VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro správu faktur v HTML5

Invoicing system in HTML5

2016 **David Hýl**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **David Hýl**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Systém pro správu faktur v HTML5
Invoicing System in HTML5

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit HTML5 aplikaci pro generování a správu faktur. Aplikace bude plně funkční i bez přístupu k internetu. Serverová část aplikace bude fungovat jako webová služba pro synchronizaci dat mezi zařízeními uživatele. Faktury bude možné exportovat do formátu PDF. Aplikace bude napsána v některém z jazyků, jenž lze překládat do JavaScriptu. Součástí práce bude přehled a srovnání těchto jazyků.

Hlavní body zadání:
1. Přehled existujících řešení.
2. Přehled a srovnání jazyků, jenž lze překládat do JavaScriptu.
3. Návrh a implementace webové služby pro synchronizaci dat mezi zařízeními.
3. Návrh a implementace offline HTML5 aplikace pro generování a správu faktur.
4. Uživatelské testování.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] PILGRIM, Mark. HTML5: up and running. 1st ed. Sebastopol, CA: O'Reilly, 2010. ISBN 05-968-0602-7.
[2] MAHARRY, Daniel. TypeScript revealed.: Apress, 2013, xix, 83 pages. Expert's voice in .NET. ISBN 1430257253.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Janoušek**

Datum zadání: 01.09.2014
Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *30. června 2016*

.....
podpis studenta

Poděkování

Rád bych poděkoval Ing. Janovi Janouškovi za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Tato práce popisuje aktuálně existující fakturační systémy, které používají k prezentaci primárně webové rozhraní a webové technologie. Zároveň tato práce popisuje a srovnává nejpoužívanější programovací jazyky, které jsou následně překládány do JavaScriptu. Další částí je vlastní návrh a implementace fakturačního systému skládajícího se z webové služby pro synchronizaci dat mezi zařízeními a HTML5 aplikace pro generování a úpravu faktur. Závěrem tato práce představuje postup a následné výsledky uživatelského testování vytvořené aplikace.

Klíčová slova

fakturační systém, HTML5, CSS3, TypeScript, JavaScript, transcompiler, offline, webová aplikace, webová služba

Abstract

This thesis describes existing invoicing systems, which use WEB interface for presentation and WEB technologies. Also this thesis describes and compares the most used programming languages, which are translated to JavaScript. Next part is my own design and implementation of invoicing system containing WEB service for synchronization of data between multiple devices and HTML5 application used for generating and editing invoices. Finally this thesis presents testing process and results based on user testing.

Key words

invoicing system, HTML5, CSS3, TypeScript, JavaScript, transcompiler, offline, web application, web service

Obsah

Seznam použitých zkratek

Zkratka	Význam
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP	Active Server Pages
CSS	Cascading Style Sheets
DOM	Document Object Model
DPH	Daň z přidané hodnoty
EU	Evropská unie
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IIS	Internet Information Serivces
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model View Controller
OS	Operační systém
PDF	Portable Document Format
QR	Quick Response
SP	Service Pack
SQL	Structured Query Language
SSL	Secure Sockets Layer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VBA	Visual Basic for Applications
VM	Virtual Machine
WWW	World Wide Web

Seznam ilustrací a seznam tabulek

Úvod

Tato bakalářská práce si klade za cíl představit doklady, které jsou běžně používané jako potvrzení o prodeji zboží či služeb. Tyto doklady se nazývají faktury. Co vlastně je faktura a jaké musí mít náležitosti, aby byla platná, je otázkou kterou tato práce zodpovídá již v první kapitole. Faktury se však v dnešní době více než v podobě papírové vytvářejí v elektronické podobě. Proto se začaly vytvářet takzvané fakturační systémy. Co jsou tyto systémy, k čemu slouží a jaké jsou mezi nimi rozdíly, je dalším tématem této práce.

Výsledkem této bakalářské práce je však nejen popis a srovnání fakturačních systémů, ale i návrh vlastního fakturačního systému. Zbývající části této práce se zabývají především způsobem, jakým je fakturační systém, který byl vytvořen v rámci této práce, navržen a následně zpracován. Na závěr byl tento fakturační systém otestován několika náhodnými uživateli.

1 Přehled existujících řešení

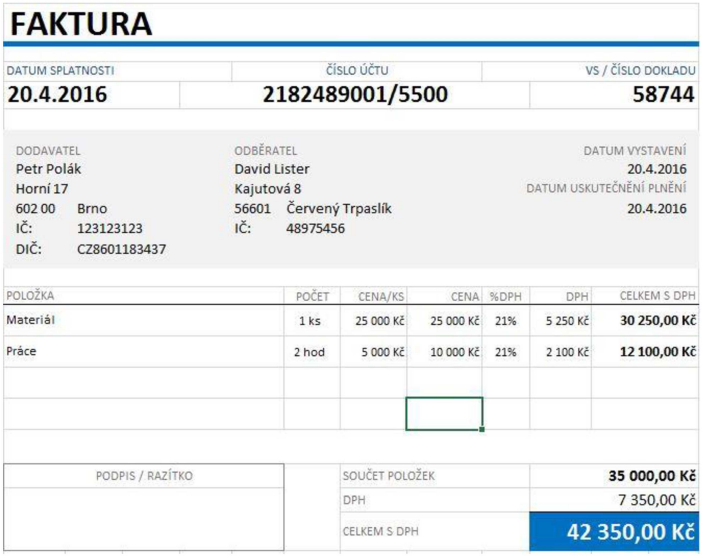
Tato kapitola se bude věnovat tomu, jaké náležitosti musí splňovat dokument, aby se stal plnohodnotnou fakturou, dále vysvětlí co je to fakturační systém, jaké jsou jeho výhody a nevýhody a jaké existují typy fakturačních systémů. V další části kapitoly bude následovat přehled již existujících fakturačních systémů. Budou zde představeny jejich přednosti a slabiny. Závěrem kapitoly budou vybrané fakturační systémy porovnány a bude z nich vybrán ideální zástupce.

1.1 Faktura

Faktura (z latinského facere – dělat, udělat) je účet za provedenou práci nebo dodané zboží. Obvykle obsahuje popis toho, za co se má zaplatit, způsob platby a datum splatnosti. Pojem faktura jako takový není definován v žádném právním předpisu. Přesto je pojem faktura vžitý a používá se běžně v obchodním styku. Také náležitosti faktury jsou ustálené tak, aby vyhovovaly podmínkám různých zákonů [1].

Každá faktura musí obsahovat následující náležitosti:

- Označení Faktura (v případě, že fakturujete s DPH je označení Faktura nepovinné, ale musí být uvedeno označení Daňový doklad)
- Jméno a adresu toho, kdo ji vydal
- Jméno (název firmy) a adresu toho, komu je určena
- Datum vydání a datum splatnosti
- Pokud je faktura určena plátci DPH, musí obsahovat datum zdanitelného plnění
- Pořadové číslo, které musí být jedinečné
- Podpis vystavovatele faktury a razítko není nutné (podpisem fakturu opatří příjemce při jejím zaúčtování)
- Faktura vystavená v elektronické podobě může být opatřena digitálním podpisem [2]



FAKTURA

DATUM SPLATNOSTI	ČÍSLO ÚČTU	VS / ČÍSLO DOKLADU
20.4.2016	2182489001/5500	58744

DODAVATEL	ODBĚRATEL	DATUM VYSTAVENÍ
Petr Polák	David Lister	20.4.2016
Horní 17	Kajutová 8	DATUM USKUTEČNĚNÍ PLNĚNÍ
602 00 Brno	56601 Červený Trpaslík	20.4.2016
IČ: 123123123	IČ: 48975456	
DIČ: CZ8601183437		

POLOŽKA	POČET	CENA/KS	CENA	%DPH	DPH	CELKEM S DPH
Materiál	1 ks	25 000 Kč	25 000 Kč	21%	5 250 Kč	**30 250,00 Kč**
Práce	2 hod	5 000 Kč	10 000 Kč	21%	2 100 Kč	**12 100,00 Kč**

PODPIS / RAZÍTKO	SOUČET POLOŽEK	**35 000,00 Kč**
	DPH	7 350,00 Kč
	CELKEM S DPH	**42 350,00 Kč**

Obrázek 1.1: *Vzorový příklad faktury obsahující všechny potřebné náležitosti*

1.2 Co je fakturační systém

Fakturační systém je zpravidla aplikace, která obsahuje přehledné uživatelské rozhraní, ve kterém umožňuje provádět veškeré činnosti s fakturami. Stará se o vytváření nových faktur, editaci uložených faktur, obsahuje možnosti archivace a exportu faktur. Účelem takového systému je usnadnit uživateli práci s fakturami. Bez takového systému je nutné faktury vytvářet ručně vyplněním předpřipravené šablony pro fakturu. Tu je následně nutné v papírové kopii uschovat, což vede ke zvýšeným požadavkům na prostor a zároveň také s sebou nese určité finanční zatížení. Tyto dva negativní aspekty je možno snížit či dokonce zcela eliminovat za použití elektronického fakturačního systému.

Fakturační systém je navržen tak, aby uživateli pomáhal při práci s fakturami. Například umožňuje uživateli uložit si zákazníky do databáze, čímž urychluje vyplnění faktury v případě opakovaného vystavování faktur jednomu zákazníkovi. Při vytváření faktur si systém sám hlídá, zda uživatel nepoužívá duplicitní číslo faktury. Zároveň uživatel nemusí vyplňovat údaje o sobě, fakturační systém si je do každé faktury doplní sám. Mezi další vlastnosti fakturačního systému patří automatická kalkulace ceny faktury z vložených položek a zadaného DPH.

Dále lze ve fakturačním systému nalézt přehledovou stránku, která informuje uživatele například o počtu vystavených faktur, o celkové sumě, za kterou byly faktury vystaveny a další vybrané statistiky. Dokáže tyto informace přehledně rozdělit například do časových úseků, a také je vyexportovat, což může být vhodné například pro roční vyhodnocení aktivit uživatele s jeho fakturami.

Mezi nesporné výhody fakturačního systému patří zejména to, že jsou data uložena elektronicky. Vytváření faktur je prováděno v grafickém rozhraní, které je navrženo tak, aby předcházelo chybám, které by mohl uživatel způsobit. Jedná se především o chyby typu nesprávně vyplněného data, osobních údajů, čísel účtů, DPH a cen. Faktury není nutné skladovat ve vlastních prostorách, čímž šetří uživateli místo. Tato skutečnost platí pouze v případě, že uživatel používá digitální podpis, v tomto případě totiž musí faktury tisknout a opatřit je podpisem a razítkem.

Největší nevýhodou fakturačního systému je to, že je pro jeho provoz nutno používat počítač či jiná elektronická zařízení, jako jsou například tablet nebo chytrý telefon. V případě, že například selže počítač, tak může uživatel ztratit veškerá uložená data. Tomuto problému se dá předejít ukládáním dat do databáze běžící na serveru umístěném v internetu. Tímto však nastává další problém, kterým je selhání připojení k internetu. V tomto případě záleží na návrhu samotného fakturačního systému, zda je možné jej využívat i za předpokladu, že uživatel pracuje offline. Tohoto docílí například ukládáním dat na interní úložiště zařízení a následnou synchronizaci dat s databází v době, kdy je připojení k internetu opět k dispozici. Další možností je mít databázový server umístěný v interní síti společnosti, díky čemuž není připojení k internetu vyžadováno.

Ve výsledku je však výhodnější používat fakturační systémy, než vypisovat faktury ručně. Dnes je již běžná praxe velkých, středních i malých firem tyto systémy využívat. Tyto systémy se liší většinou svou složitostí a rozmanitostí funkcí. Od jednoduchých tabulek vytvořených v Microsoft Office Excelu, přes webové aplikace navržené na obsluhu mnoha uživateli, za poskytnutí co největší nabídky služeb, až po plně individuální řešení pro konkrétní společnosti či osoby, které mohou být například součástí jednoho velkého informačního systému, kde jeho jednotlivé součásti vzájemně komunikují.

Nejzákladnějším typem fakturačního systému je vytvoření interaktivní tabulky v programu Microsoft Office Excel. Ten umožňuje použití programovacího jazyka VBA (Visual Basic for Applications). Za jeho pomocí lze proces vytváření faktur automatizovat a nastavit ověřování vložených hodnot do formuláře. Ve formulářích lze také využívat tlačítka a vstupy jako textové pole nebo zatrhávací tlačítko. Avšak jeho možnosti jsou velmi omezené, především proto, že je vázán na program MS Excel. V dnešní době se od tohoto řešení spíše upouští. Můžeme jej nalézt především u menších firem či živnostníků, kteří si jej vytvořili nebo nechali vytvořit už před několika lety a ten jim vyhovuje dodnes.

Dalším typem fakturačního systému je takový systém, který je napsán jako webová aplikace. Tento typ je v dnešní době velice rozšířený. Většinou je jeho funkčnost napsána v programovacím jazyce JavaScript a uživatelské rozhraní ve značkovacím jazyce HTML (HyperText Markup Language), jehož vzhled je nastaven pomocí kaskádových stylů (CSS). Můžeme se také například setkat s aplikacemi napsaných v programovacích jazycích Java nebo PHP. Java podporuje také tvorbu uživatelského rozhraní, PHP jej vykresluje obvykle pomocí HTML a CSS stejně jako JavaScript.

Posledním typem fakturačního systému je takový, který je napsán v jazyce k vytváření aplikací pro operační systémy, jako jsou Windows, Linux či MAC-OS. Mezi tyto jazyky se řadí například Java, C++ nebo C#. Oproti systémům napsaným jako webová aplikace se většinou liší pouze zpracováním grafického rozhraní, na které se nepřistupuje přes prohlížeč, ale spouští se přímo v počítači. Integraci s jiným informačním systémem používaným ve firmě lze provést pomocí webové aplikace, kde k tomuto účelu slouží především poskytnuté API, které využívá firemní informační systém. V případě programu bývá fakturační systém jako takový přímou součástí firemního informačního systému, který je přímo na míru uzpůsoben uživateli.

Protože se tato práce zabývá fakturačním systémem napsaným v jazyce HTML5, tedy napsaným jako webová aplikace, bude se srovnání existujících řešení týkat pouze fakturačních systémů napsaných tímto způsobem.

1.3 Srovnání existujících fakturačních systémů v HTML5

Pro inspiraci k vytvoření vlastního fakturačního systému bylo vhodné si prohlédnout již existující řešení fakturačních systémů běžně dostupných veřejnosti. Z nich je pro účely této kapitoly vybráno několik zástupců, jejichž klady a zápory budou dále v této kapitole porovnány. Jedná se o fakturační systémy především českého původu, avšak vyskytuje se mezi vybranými fakturačními systémy také jeden původem z Polska. Vybrán byl především proto, že umožňuje vytváření faktur se všemi náležitostmi požadovanými v České republice.

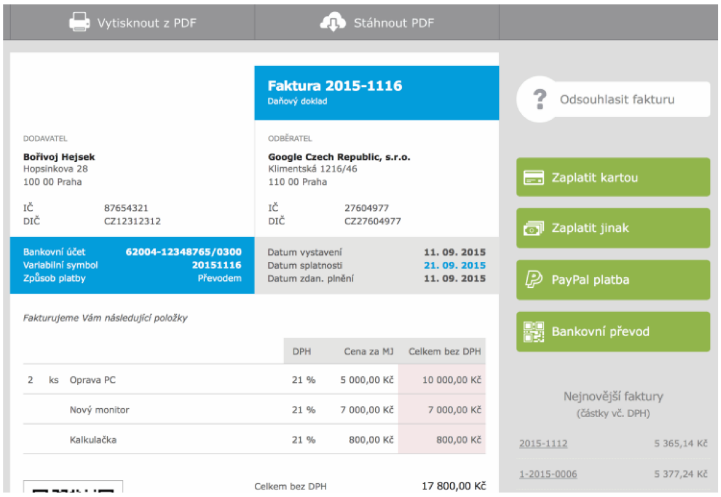
Vybrané fakturační systémy jsou:

- fakturoid.cz
- idoklad.cz
- fakturman.cz
- ifakturant.cz
- invoiceocean.com

Prvním z představených fakturačních systémů je Fakturoid. Tento systém je velice rozsáhlý a profesionálně zpracovaný. Již v neplacené verzi, která je v tomto případě omezena počtem 5 klientů,

pro které je možno faktury vystavovat, umožňuje vytváření běžných faktur, zálohových faktur a proforma faktur (pro subjekty které nejsou plátce daně). Umožňuje umístit si do faktury vlastní logo společnosti i podpis. Výslednou fakturu lze následně vyexportovat do PDF. Zároveň nabízí vlastní mobilní aplikaci pro iPhone a Android. Další nabízenou službou je vlastní API, přes které je možno přistupovat k databázi faktur skrze vlastní aplikaci (například mobilní bankovnictví).

Co se cenových tarifů týče, tak mimo základní tarif nabízí Fakturoid ještě další tři tarify. Tarif "Sólo", který přidává možnost fakturovat pro neomezený počet klientů, párování plateb s bankovním účtem, generování daňového přiznání a použití 4 vzhledů faktur. Tarif "Sólo+" přidává možnost pravidelného fakturování, integraci PayPal plateb, rozšiřuje počet povolených požadavků z API na 6 000 požadavků za jeden měsíc. Dále přidává možnost změny barev faktury a možnost vygenerovat přiznání k DPH, kontrolní hlášení a souhrnné hlášení k DPH pro EU. Posledním tarifem je tarif "Firma", ten umožňuje do fakturačního systému přístup až 5 uživatelům, kde každý může mít jiná oprávnění. Další možností tohoto tarifu je, že umožňuje využívat platební bránu GoPay, spárovat platby s více než jedním bankovním účtem a rozšiřuje počet API požadavků za měsíc na 50 000 požadavků za měsíc.

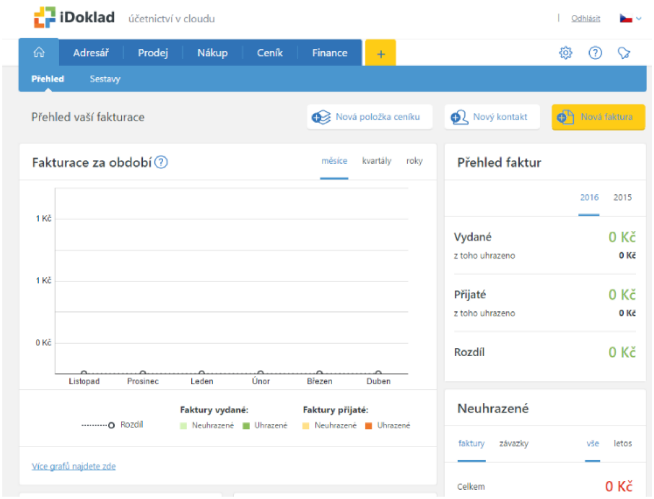


Obrázek 1.2: *Ukázka uživatelského rozhraní Fakturoidu*

Dalším představeným fakturačním systémem je iDoklad. Ten okamžitě po vstupu na stránku zaujme zajímavým vzhledem, ten však může i uškodit, například při připojování přes mobilní síť bude načtení stránky mnohem míce náročné na objem přenesených dat. Stránka totiž k upoutání pozornosti mimo obrázků využívá taká videa běžícího v pozadí. Opět je vidět, že fakturační systém byl navržen velice profesionálně. Fakturační systém je jednoduchý k ovládání a jeho vzhled vypadá moderně. Veškeré faktury jsou ukládány do cloudu, což zaručuje velmi nízké riziko ztráty dat. Největší předností iDokladu je cena. Fakturační systém iDoklad je totiž provozován zcela zdarma bez poplatků za cokoliv. Tato skutečnost je však vynahrazena vysokým množstvím reklam, které se vyskytují jak v samotném fakturačním systému, tak zároveň i v emailových zprávách. Reklama je pro autory tedy jediným, nebo velmi výrazným způsobem jak z něj generovat zisk.

Co se funkční stránky týče, nabízí fakturační systém iDoklad možnost vystavování běžných faktur. Ty lze vytvořit výběrem z mnoha českých i cizojazyčných šablon. Dále je na faktury možné umístit vlastní logo, podpis i razítko. Do faktury přidává také QR kód pro možnost rychlého vyhledání a zobrazení faktury, především při použití mobilní aplikace.

Další z funkcí je integrace s internetovým bankovnictvím, díky čemuž dokáže iDoklad například přehledně zobrazit, kdo své faktury ještě nezaplatil, případně označit již zaplacené faktury okamžitě po připsání peněz na účet. Umožňuje fakturovat i v cizích měnách, vždy podle pravidel daného státu, včetně režimu přenesení daňové povinnosti. Umožňuje nastavit zákazníkům automatické i jednorázové slevy. Zajímavou službou je ověření každého zákazníka v databázi Creditcheck, tato služba porovnává velké množství registrů neplatičů v ČR, dále prochází například aukční portály, či obchodní rejstřík. Díky této služby může uživatel okamžitě vědět, zda jeho zákazník bude schopen objednávku uhradit a předejít tak možným problémům. Mezi další funkce patří možnost opakovaného vystavení faktur a automatické upomínky plateb zákazníkům. Také nabízí mobilní aplikaci pro tři nejrozšířenější mobilní platformy Apple iOS, Android a Windows Mobile.



Obrázek 1.3: *Ukázka uživatelského rozhraní iDoklad*

Fakturační systém iFakturant je vzhledově velmi jednoduchý a přehledný. Mezi své hlavní přednosti staví právě jednoduchost a bezpečnost, kterou zajišťuje SSL šifrováním. Obsahuje pouze jediný tarif, jeho cena se odvíjí pouze od doby předplacení. Verze zdarma vyprší po 10 dnech. Vytvářet lze klasické faktury i proforma faktury. Umožňuje vložit do faktur vlastní logo i podpis. Lze nastavit text automatického emailu, jak pro zákazníka, tak pro účetního. iFakturant umožnuje ukládat položky do ceníku, které usnadňují práci s položkami. Lze si také vytvořit seznam zákazníků a ceník zboží s možností sledování množství na skladě. Množství zákazníků, faktur ani položek ceníku není nijak omezeno. Dále lze vytvořit opravný daňový doklad, automaticky vypočítat slevu z faktury a odeslat upomínku zákazníkovi.

Fakturaman je jedním z jednodušších fakturačních systémů, to znamená, že neobsahuje tolik funkcí jako prozatím představené fakturační systémy. Vzhled je jednoduchý a čistý. Umožňuje vytvářet běžné faktury, faktury bez DPH, zálohové faktury i proforma faktury. Umožňuje fakturovat v různých měnách a jazycích. Dále lze do faktury přidat vlastní razítko, logo i podpis. Tyto funkce jsou pro fakturování většinou postačující, avšak existují i neplacené fakturační systémy, které zvládnou toto a mnohem více. Existují pouze dva tarify "Základní" a "Rozšířený", které se liší pouze absencí historie faktur v základním tarifu. Verze bez placení neexistuje.

InvoiceOcean je zástupce zahraničních fakturačních systémů, konkrétně z Polska. Jeho vzhled je moderní a čistý. Jeho uživatelské rozhraní nepodporuje český jazyk, toto by určitě mohlo některé případné uživatele z České Republiky odradit od jeho používání. Většinu důležitých jazyků však podporuje, patří mezi ně například Angličtina, Polština, Němčina nebo Francouzština. Výsledné faktury však lze vytvářet i v českém jazyce. Proto je možné jej srovnávat i s českými fakturačními systémy.

Cena za tento fakturační systém určuje množství funkcí, které může uživatel využít. Verze "Free", která je zdarma umožňuje vytvořit pouze 3 faktury. Podporuje vytváření faktury ve všech měnách a následné odeslání faktury zákazníkovi přes email. Všechna data ukládá do cloudu, což znamená, že jsou data uložena v bezpečí a dostupná odkudkoliv. Platby jsou propojeny s platebními systémy PayU a PayPal, toto velmi usnadňuje fakturování v případě platby faktury kartou (PayU) a platby přes službu PayPal. Do faktur lze vložit vlastní logo a použít různé šablony faktur. Faktury lze také exportovat do PDF a tím si je uchovat v tisknutelné podobě jinde, než jen na serverech, které využívá InvoiceOcean. Další verzí je "Basic", ta již umožňuje vystavit neomezené množství faktur, zobrazit statistiky a nastavit připomínky. Faktury lze vytvářet ve všech 29 dostupných jazycích, mezi ně patří například Čeština, Francouzština, Španělština, Polština, nebo Albánština. Fakturační systém je také možno integrovat do vlastního e-shopu a k přístupu k vlastním fakturám použít předpřipravené API.

Verze "Professional" je již téměř bez omezení a využívá všech dostupných možností fakturačního systému. Přidává možnost vytvořit si vlastní šablony pro faktury, emaily a dokonce změnit vzhled samotného systému. Přidává také možnost automatických plateb, opakování faktur a rozšířené statistiky. Učet může využívat více uživatelů v jednom účtu (maximálně 5), což se hodí pro menší firmy, které chtějí fakturovat pod jedním účtem, ale již mají více než jednoho člověka pověřeného financemi firmy. Lze také nastavit automatické platby a umožňuje správu skladů, kde přehledně zobrazí kolik zboží je ještě na skladě a automaticky jej upraví v případě vystavení faktury. Poslední verze "Enterprise" již nepřidává žádnou zásadní funkci, pouze umožňuje využívat bez omezení všechny předchozí funkce.

1.4 Výsledek srovnání

Po srovnání vybraných fakturačních systémů se ukázalo, že na trhu je velká konkurence a je tedy nutné případným zákazníkům nabídnout něco navíc. Rozdíly mezi těmi více a méně propracovanými fakturačními systémy jsou docela velké, záleží však na požadavcích každého uživatele, respektive na množství faktur, které potřebuje fakturovat, kolik je za tuto službu ochoten zaplatit a jaké nadstandartní služby chce využívat.

Fakturační systém Fakturaman nabízí nejméně funkcí a jeho cena je v porovnání s ostatními docela vysoká. Vzhledem k množství funkcí jej nelze doporučit ani pro uživatele, kteří si přejí fakturovat jen velmi málo. iFakturant opět nenabízí mnoho funkcí oproti ostatním a zároveň je jeho cena ještě vyšší než u Fakturamana, takže jej také nelze doporučit a pro srovnání zbývají tedy pouze 3 fakturační systémy. Jedním z nich je fakturační systém InvoiceOcean, na něm je vidět že si na něm dali tvůrci více záležet, ale absence českého jazyka v systému a také vyšší cena jej opět zařazují do kategorie "nedoporučuji".

Zůstaly tedy dva ryze české fakturační systémy - Fakturoid a iDoklad. Oba nabízí velké množství funkcí, které dokáží fakturování co nejvíce zjednodušit a udělat přehlednějším. Pro menší firmy a živnostníky by byl vhodný fakturační systém iDoklad, především v případě, že si přejí za fakturování neutrácet peníze, avšak to s sebou nese nutnost akceptovat přítomnost reklam, jak v systému

samotném, tak i v emailech, které odesílá. Vítězem tohoto srovnání je tedy fakturační systém Fakturoid. Ten nabízí veškeré potřebné funkce pro pohodlné a rychlé fakturování. Zároveň umí generovat daňové přiznání, integraci s účetními systémy, vlastní API pro e-shopy i tarif "Firma", který umožňuje přístup k fakturám více lidem v rámci jedné firmy. Je vhodný pro kohokoliv, kdo chce mít přehled ve svých fakturách. Jeho cenové tarify jsou opravdu velmi příjemné. Nabídne za to obrovské množství funkcí, které předčí veškerou svou konkurenci.

Tabulka 1.1: *Přehledné srovnání funkcí srovnávaných fakturačních systémů*

	Fakturoid	iDoklad	Fakturman	iFakturant	InvoiceOcean
Běžné faktury	✓	✓	✓	✓	✓
Proforma faktury	✓	✖	✓	✓	✓
Vlastní logo	✓	✓	✓	✓	✓
PDF	✓	✓	✓	✓	✓
Párování s bank. Účtem	✓	✓	✖	✖	✖
PayPal	✓ **Sólo+**	✖	✖	✖	✓
Upomínky	✓	✓	✖	✓	✓ **Basic**
Daňové přiznání	✓	✓	✖	✖	✖
Mobilní aplikace	✓	✓	✖	✖	✖
API	✓	✓	✖	✖	✓
Cena	**Zdarma** 5 klientů	**Zdarma**	**Základní** 30Kč	**3 měsíce** 417Kč	**Free** 3 faktury
	Sólo 160Kč		**Rozšířený** 80Kč	**6 měsíců** 774Kč	**Basic** $9
	Sólo+ 320Kč			**12 měsíců** 1428Kč	**Professional** $18
	Firma 640Kč				**Enterprise** 32$

2 Přehled jazyků, které lze překládat do JavaScriptu

Tato kapitola odpoví na otázku co je programovací jazyk JavaScript, k čemu slouží jazyky, které lze následně překládat do JavaScrpitu, jací jsou zajímaví zástupci těchto jazyků a jaké jsou mezi nimi rozdíly. Dále jsou tyto jazyky srovnány mezi sebou a výsledkem je doporučení jaké jazyky používat, a kterým se raději vyhnout.

2.1 Co je JavaScript

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, jehož autorem je Brendan Eich z tehdejší společnosti Netscape.

Nyní se zpravidla používá jako interpretovaný programovací jazyk pro WWW stránky, často vkládaný přímo do HTML kódu stránky. Jsou jím obvykle ovládány různé interaktivní prvky GUI (tlačítka, textová políčka) nebo tvořeny animace a efekty obrázků.

Jeho syntaxe patří do rodiny jazyků C/C++/Java. Slovo Java je však součástí jeho názvu pouze z marketingových důvodů a s programovacím jazykem Java jej vedle názvu spojuje jen podobná syntaxe.

Program v JavaScriptu se obvykle spouští až po stažení WWW stránky z Internetu (tzv. na straně klienta), na rozdíl od ostatních jiných interpretovaných programovacích jazyků (např. PHP a ASP), které se spouštějí na straně serveru ještě před stažením z Internetu. Z toho plynou jistá bezpečností omezení, JavaScript například nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele.

2.2 Jazyky, které se následně překládají do JavaScriptu

Jazyky, které se následně překládají do JavaScriptu mají jediný cíl a tím je usnadnit uživateli psaní kódu v JavaScriptu a dodat mu některé možnosti, které čistý JavaScript nemá. Docílí toho především jinou, často jednodušší syntaxí podobnou jinému známému programovacímu jazyku a přidává například kontrolu datových typů. Kód se po zkompilování přeloží tak, aby odpovídal normě ECMAScript, která udává to jakou syntaxi má JavaScript využívat. Z toho tedy vyplývá, že díky těmto jazykům můžeme psát kratší a přehlednější kódy, u kterých máme jistotu, že se vždy přeloží do správné syntaxe JavaScript kódu. Další výhody jazyků jsou specifické pro každý z nich.

Jazyků, které se následně překládají do JavaScriptu, je velké množství, pro srovnání jsou zde vybrány pouze ty nejzajímavější a nejrozšířenější. Na několika následujících stránkách budou srovnány tyto jazyky:

- TypeScript
- GorillaScript
- CoffeScript
- Dart
- Script#

TypeScript je jako jediný z těchto jazyků nadstavbou jazyku Javascript. TypeScript injektuje vlastní kód jedině, když je použita dědičnost. I tak se ale jedná pouze o 6 řádků kódu navíc. Tento jazyk

nabízí kontrolu typů, umožňuje využívání tříd, modulů nebo rozhraní. Podporuje také IntelliSense a refaktorizaci. Jeho syntaxe je navržena tak, aby byla co nejvíce shodná se syntaxí ECMAScript 6. TypeScript tedy nabízí statickou typovou kontrolu.

Proč se učit TypeScript a nepočkat, až bude někdy v budoucnu v JavaScriptu? Protože uvedení typů bylo součástí specifikace ECMASCript 4, ale výsledkem bylo, že od toho každý dával ruce pryč. Nakonec tedy vznikla odlehčená specifikace ECMAScript 6, která typy neobsahuje. V současné době je typová kontrola pro JavaScript v nedohlednu.

Práce s programovacím jazykem TypeScript je mnohem jednodušší než s čistým JavaScriptem především díky tomu, že je velice podobný jazykům jako C++/C#, ale zároveň umožňuje využít čistý JavaScript kód, což je jeho obrovská výhoda. V případě, že je třeba vyřešit nějaký specifický problém, je možné mnohokrát nalézt odpověď právě v programovacím jazyce JavaScript, a tu lze bez problému využít. Na oficiálních stránkách TypeScriptu existují přehledné příklady práce s kódem a zároveň s ním pracuje početná komunita, takže dohledání informací, když některá část kódu nefunguje, není většinou žádný problém.

Dále se podíváme na programovací jazyk Gorilla Script. Tento jazyk se zaměřuje na prevenci chyb, kterou čistý JavaScript není schopen odhalit. Například při sčítání různých datových typů ohlásí chybu, pokud je není možné sečíst. JavaScript by například pouze sečetl dvě řetězcové hodnoty. Přidává možnost využívat konstanty. Není nutné používat složené závorky "{}", kód je schopen sám rozeznat konec části kódu pomocí odsazení, většina operátorů je v něm pozměněna, například *"*===" -> "==*"*, nebo *"&&*" -> "*and"* a přidává několik nových syntaxí matematických funkcí *"x^y*" -> *"Math.pow(x, y)", nebo "x min= y" -> "pokud x je menší než y tak x=y"*.

Umožňuje práci s řetězci, tak že proměnnou lze definovat znakem "$", například *"Jméno: $jmeno"*, zároveň povoluje víceřádkové řetězce. Přidává do podmínky "if" možnost "unless", která slouží jako opak příkazu "if". Zjednodušuje syntaxi cyklů, přidává možnosti "to", "til" a "by". Nabízí možnost ověření, zda proměnná existuje a má hodnotu jednoduše za použití symbolu "?".

GorillaScript také implementuje třídy nebo konstrukce switch a try-catch. Zjednodušuje práci s regulérními výrazy změnou syntaxe. Podporuje genericitu, globální proměnné nebo get a set pro proměnné v objektu a spoustu dalších funkcí. Z již zmíněných příkladů je patrné, že tedy do jisté míry pozměňuje syntaxi JavaScriptu a nahrazuje ji svou vlastní, což pro vývojáře znamená naučit se nový jazyk, který je stále více podobný JavaScriptu než některému z jazyků jako jsou C# nebo Java.

Dalším ze srovnávaných programovacích jazyků je CoffeeScript. Je to prakticky vzato nadstavba nad JavaScriptem, která uživatelům JS přináší spoustu „syntaktického cukru“, inspirovaného Pythonem a Ruby. Nabízí čistší funkcionální zápis a zajišťuje, že výsledný kód používá správné konstrukce. Zápis může připadat někomu komplikovaný, jinému naopak krásně čistý.

CoffeeScript (dále pouze CfS) se inspiroval u Pythonu a používá stejný způsob zápisu bloků pomocí odsazení. Úvodní mezery na řádku jsou tedy důležité, označují zanoření bloku. CfS nemá složené závorky k označování bloků, místo nich použijete odsazení. Stejně tak není nutné psát středníky za příkazem, výrazem či funkcí. Středník použijete pouze v případě, že chcete na jednom řádku uvést např. více volání funkcí.

U volání funkce s parametry není nutné použít závorky – parametry prostě následují za jménem funkce: *print "Ahoj"* a pokračují až do konce řádku nebo bloku. Volání funkce bez parametrů je potřeba

zapsat se závorkami. Řetězce se v CfS zapisují jako v jiných jazycích, tj. do uvozovek či apostrofů. Podobně jako v PHP zde platí, že řetězec zapsaný v apostrofech je konstantní, v řetězci zapsaném v uvozovkách proběhne nahrazení proměnných a výpočet výrazů, pokud jsou zapsané jako #{...}. Zápis objektových literálů či polí je podobný JSON – tedy objekty se zapisují do složených závorek, pole do hranatých.

Příkaz "if" lze zapisovat bez závorek okolo podmínky a bez složených závorek okolo bloku (víceřádkové bloky se opět vyznačují pomocí odsazení). Nahrazuje operátory podobně jako GorillaScript. Nabízí strukturu class, v níž lze určit jméno třídy, třídu nadřazenou, přiřadit vlastnosti a nadefinovat konstruktor. Tento jazyk také upravuje a zjednodušuje syntaxi a bude nejvíce sedět programátorům zvyklým na jazyky Python a Ruby.

Programovací jazyk Dart je opensource platforma pro vývoj HTML5 webových aplikací vytvořená firmou Google. Dart je více než jen jazyk - patří k němu také vlastní knihovny, editor, virtuální stroj (VM, po vzoru Javy), prohlížeč Dartium, který je schopný spustit Dart nativně s mnohem vyšším výkonem a kompilátor do JavaScriptu (dart2js).

Dart obsahuje třídy, dědičnost a celkově je syntaxe velmi podobná C# nebo Javě. Podporuje také znuvupoužitelnost kódu, což zajišťuje možnost exportovat knihovny pomocí vlastního package manažera (pub). Vývojáři napsali také spoustu kvalitních knihoven, například pro práci s datovými strukturami, matematikou, kódování URL, kryptografii, DOM a spoustu dalších.

V Dartu je taky možné využít již existující JavaScriptový kód pomocí knihovny dart.js, díky čemuž může využívat i knihovny napsané pro JavaScript. Jeho editorem je upravený Eclipse, kde je možné debuggovat kód, podporuje refactoring, IntelliSense i detekci potenciálních chyb. Dart má také dobrou podporu pro asynchronní kód. Používá pro to "futures", které reprezentují výsledek, který bude spočítán někdy v budoucnosti. To umožňuje napsat asynchronní kód přehledně. Dart také podporuje vícevláknost, kde pomocí "isolates" umožňuje napsat bezpečný vícevláknový kód.

Samotná syntaxe kódu je nejvíce podobná zápisu programovacího jazyka Java, a každý, kdo v něm chvíli pracoval, nebude mít s Dartem větší problémy.

Programovací jazyk Script# je framework, který umožňuje překládat kód C# do JavaScriptu. Kód překládá takovým způsobem, jako by byl psán ručně, takže následná editace a debugging je přehledný. Další výhodou Script# je podpora IntelliSense, detekce chyb při kompilaci, možnost vytvářet dokumentaci, refactoring a debugging.

Ideálním editorem je tedy Microsoft Visual Studio, do kterého stačí pouze nahrát Script# plugin a práce může začít. S novějšími verzemi je však Script# méně stabilní a náchylnější k chybám za běhu. Microsoft jeho starší verze využíval k naprogramování velkých webových aplikací, jako například Office Web Apps, Hotmail, Office 365, nebo SharePoint 2013. Tento programovací jazyk je tedy především vhodný pro programátory, kteří mají velmi silně zažité programování v C# a nechtějí se učit odlišných syntaxí jiných jazyků. Pro větší firmy je například výhodnější použít Script# než organizovat školení pro programátory na zcela jiný jazyk.

2.3 Závěrečné srovnání jazyků

Každý z vybraných programovacích jazyků vytváří JavaScript soubor (.js), ale každý z nich se k němu dostane trošku jinou cestou. Většina z nich nějakým způsobem podporuje kontrolu typů, ale

zdaleka nejlépe se s touto problematikou popasoval jazyk TypeScript, který využívá statické typy. Tím dokáže svou syntaxi velmi přiblížit jazykům jako C# nebo Java, a zároveň nabídnout příjemné prostředí editoru podpořené IntelliSense, refactoringem a detekcí potencionálních chyb.

Tyto možnosti nabízí také programovací jazyk Dart. Vývojáři Googlu si zvolili zcela jinou cestu jak vytvořit jazyk přeložitelný do JavaScriptu. Vytvořili zcela novou platformu, která je podobná Javě. Největší podobnost je právě v použití Virtuálního Stroje (VM), díky čemuž dokáže kód běžet kdekoliv, kde je nainstalován VM. Zároveň vytvořili i nový prohlížeč, který bude schopen efektivněji zpracovat kód přímo z Dartu namísto přeloženého kódu do JavaScriptu. V tomto jazyce by mohla být budoucnost, avšak zatím je třeba ještě zapracovat na vývoji a pokusit se tuto platformu rozšířit na co nejvíce zařízení. Velká spousta webových vývojářů už má zažitý, jak čistý JavaScript, tak jeho nadstavby. Dart si s nimi dokáže poradit, ovšem pro vývojáře je zatím nezajímavé využívat Dart, když mají vše potřebné již v JavaScriptu.

Gorilla Script se snaží ulehčit práci a zvýšit přehlednost kódu tím, že mění syntaxi JavaScriptu. Takže pokud se vývojář dokáže tuto syntaxi naučit, nebude muset tolik psát. Přidává několik běžně používaných konstrukcí v jazycích jako Java a C#, například *switch* a *try-catch*. Také obsahuje podporu tříd, k čemuž připadají také konstruktory a destruktory. Proměnné je možno zapisovat také pomocí *get-set* jako property. Výsledný kód tohoto jazyka je kombinací mezi jazyky JavaScript a TypeScript. TypeScript se však jeví pro vypracování této bakalářské práce jako vhodnější kandidát.

CoffeScript je jazyk, který svým zápisem nejvíce připomíná Python, namísto závorek používá odsazení a také pozměňuje syntaxi JavaScriptu, aby byl kratší, podobně jako Gorilla Script. Také obsahuje možnost práce s třídami. Tento jazyk je dobrou volbou pro programátory, kteří jsou zvyklí na programovací jazyk Python. Pokud se chce programování webových aplikací věnovat vývojář tímto jazykem nezasažený, měl by se poohlédnout po jiném programovacím jazyce.

Posledním jazykem je Script#, který je ideální volbou pro C# vývojáře, nabízí prakticky vše, co předchozí porovnávané jazyky, takže práce s ním půjde C# vývojářům velmi dobře.

Zatímco TypeScript je rozšíření JavaScriptu, CoffeeScript je odlišný jazyk a není typový. Nemůžete například vzít své JavaScriptové soubory, přejmenovat je na CoffeeScript a rozšiřovat je. Dart nabízí odlišné prostředí než JavaScript, ale za cenu toho, že se v JavaScriptu musí uměle vytvářet. To výrazně snižuje výkon. Script# je na tom podobně.

Výsledkem tohoto srovnání je, že velmi záleží na tom, s jakým jazykem je programátor zvyklý pracovat. Pokud s Javou, měl by využít jazyk Dart, pokud byl jeho oblíbený jazyk Python, tak jeho volbou bude CoffeScript, C# vývojář zase sáhne po programovacím jazyku Script# a vývojáři, kteří už mají praxi s JavaScriptem by měli určitě sáhnout po jazyku TypeScript. Také jej lze doporučit tomu, kdo začíná s programováním webových aplikací, protože je nejlépe optimalizovaný pro překlad do JavaScriptu při zachování co nejpřesnější podpory specifikace ECMAScript 6. Programovací jazyk Gorilla Script nenabízí oproti konkurenčním jazykům žádné výhody, bylo by vhodné zvolit jiný z představených programovacích jazyků.

3 Návrh webové služby pro synchronizaci dat

3.1 Použitá architektura a technologie

Zadání bakalářské práce nijak nevymezuje to, jaké technologie by měly být použity pro implementaci webové služby. Základem pro tuto službu je místo, kde budou uloženy data. Protože se jedná o webovou službu, je nutné, aby data nebyla ukládána u uživatele, ale aby byla dostupná odkudkoliv na jakémkoliv zařízení. Většina potřebných dat je uložena v databázi a samotná webová aplikace následně na lokálním úložišti. Lze také webovou aplikaci přenést na webový server, tak aby byla dostupná i přes Internet.

Součástí webové služby je také rozhraní pro komunikaci mezi serverem a samotnou aplikací. Toto rozhraní slouží jak k přijímání metod HTTP, jako jsou GET, POST a DELETE , ta také slouží ke komunikaci s aplikací na serveru, ze které volá metody, které vykonají požadovanou operaci.

Aplikací běžící na serveru pro uchování dat bude v případě této práce databázovým řešením od Microsoftu pojmenované Microsoft SQL server (dále jen MSSQL). MSSQL je relační databázový systém. Relační databáze je založena na tabulkách, kde jednotlivé řádky reprezentují jednotlivé záznamy a sloupce pak znázorňují různé atributy. Ty musí mít definovány datové typy. Jeden ze sloupců je tzv. primární klíč, který je vždy unikátní, a podle kterého se dá jednoznačně vybrat každý záznam v tabulce. Dále může být jeden nebo více sloupců takzvaným cizím klíčem, který je vždy primárním klíčem v tabulce, se kterou má tabulka obsahující cizí klíč vazbu.

Pro implementaci rozhraní byl vybrán ASP.NET Web API. Je pro něj již nachystaná kompletní šablona v Microsoft Visual Studiu, která obsahuje potřebné HTTP metody. Samotná implementace je díky tomu velmi usnadněna.

Vzhledem k tomu, že aplikace je rozdělena na dvě části, webovou službu a webovou aplikaci, nabízí se použití architektury MVC (Model, View, Controller). Každá z těchto částí plní svou důležitou roli v aplikaci, ale je možné je využít i odděleně. To znamená, že na sobě nejsou nijak závislé, pouze spolu spolupracují.

Model slouží k definici všech tabulek z databáze do objektů v aplikaci. Jsou v něm uloženy informace o všech proměnných a datových typech, a zároveň jsou v mé implementaci do modelu uloženy i délky/velikosti jednotlivých datových typů tak, aby odpovídaly tomu, co je nadefinováno v databázi. Příkladem může být atribut se jménem "Název" a datovým typem "VARCHAR(20)". Číslo v závorce udává právě maximální velikost/délku záznamu v atributu. V tomto případě je do modelu uložena konstantu s informací o tom, jak velký může maximálně být atribut "Název" a tím je dosažena konzistence dat.

Controller je rozhraním pro komunikaci mezi Webovou aplikací a serverem. V případě spuštění ASP.NET programu ve Visual Studiu je jako webový server využíván IIS Express. IIS Express je ideálním webovým server na testování webových aplikací. Jedná se o odlehčenou verzi webového serveru IIS vytvořeného firmou Microsoft. Jeho omezení spočívá především v omezeném množství podporovaných protokolů a rozšíření. Běží pouze v režimu uživatele, nikoliv jako služba v operačním systému a vyskytuje se až od verze Visual Studio 2010 SP1.

View je grafickým rozhraním, které uživateli poskytuje možnost ovládání webové aplikace a zobrazuje požadované výsledky. Té se bude dále věnovat až další kapitola.

3.2 Návrh databáze

Před samotnou implementací služby je nutné si promyslet si, co všechno je třeba mít uložené v databázi a jaké relace použít. Základní myšlenkou aplikace je to, že k ní bude moct přistupovat více než jeden uživatel, a každý bude mít možnost fakturovat. Uživatel se bude muset pro přístup ke svým fakturám vždy přihlásit. Z toho vyplývá, že v databázi bude muset být tabulka "Uživatel", která bude udržovat záznam pro každého registrovaného uživatele v systému. Uživatel bude následně moct vytvářet jednotlivé faktury. Takže další tabulkou bude tabulka "Faktura". Každá faktura obsahuje seznam položek a zákazníků. Je tedy třeba přidat další dvě tabulky "Zákazník" a "Položka".

Další možnost, se kterou je třeba počítat už při návrhu databáze je styl. Různé styly budou nabídnuty uživateli při vytváření faktury. Protože je vhodné mít předem definované styly faktur, které budou uživateli nabídnuty, nejjednodušším způsobem je pro ně vytvořit samostatnou tabulku v databázi. Toto ;umožní ve webové aplikaci pouze stáhnout seznam povolených stylů, které budou následně nabídnuty uživateli. Kdyby byl "Styl" pouze atributem namísto tabulky, tak by nebylo možné zjistit, jaké styly jsou povoleny a musel by tento seznam být definován až na úrovní webové aplikace, což by mohlo vést k nekonzistenci dat a chybám.

Další z tabulek, která bude mít nadefinovány hodnoty předem, bude tabulka "DPH", ta slouží pouze k definici povolených sazeb DPH pro položky. Poslední použitou tabulkou, je tabulka "Forma úhrady", která obsahuje možnosti platby faktury.

Poslední tabulkou, kterou obsahuje databáze je "Vlastní pole". Ta reprezentuje jakékoliv jiné pole přímo definované uživatelem. Jeho hodnota je uložena v textové podobě (bez kontroly datového typu) a může sloužit například jako vlastní identifikační číslo faktury, nebo jméno člověka, který vystavoval fakturu ve firmě.

Dalším důležitým bodem je určit jaké vztahy (relace) budou mezi sebou jednotlivé tabulky mít. Stěžejní tabulkou v navrhnuté databázi bude tabulka "Faktura". Ta obsahuje cizí klíč "Uživatel ID", z toho vyplývá, že uživatel může mít přiřazeno hned několik faktur a zároveň faktura může být přiřazena pouze jednomu uživateli. Stejné pravidlo platí také pro tabulku "Zákazník", cizí klíč v tomto případě bude "Zákazník ID". Tabulka "Faktura" dále obsahuje cizí klíč s názvem "Styl ID", která znázorňuje přiřazený styl k faktuře. Stejná relace platí také pro tabulku "Forma úhrady", cizím klíčem je "Forma úhrady ID".

Dále musí každá faktura obsahovat položky, proto tabulka "Položka" obsahuje cizí klíč s názvem "Faktura ID", tak aby každá položka mohla být přiřazena právě jedné faktuře. Dalším cizím klíčem tabulky "Faktura" je "DPH ID" a to za účelem přiřazení správné hodnoty DPH k položce. Poslední Tabulkou napojenou na tabulku "Faktura" je "Forma úhrady", a ta opět obsahuje cizí klíč s názvem "Faktura ID", stejně jako tabulka "Položka". Nesmíme zapomenout ještě na relaci mezi tabulkami "Uživatel" a "Zákazník". Každý uživatel totiž bude mít uložen svůj seznam kontaktů (zákazníků), pro které vystavil, nebo bude vystavovat faktury. Toho je docíleno vložením cizího klíče s názvem "Uživatel ID" do tabulky "Zákazník". Každý zákazník tedy bude patřit pouze jednomu uživateli.

Další částí databáze, kterou je třeba navrhnout, jsou atributy a jejich datové typy. V následující části textu je popsáno jejich umístění a význam. Kompletní návrh databáze je přehledně zakreslen do entitně-relačního diagramu umístěného v příloze [Příloha A].

První rozebíranou tabulkou je "Zákazník". O zákazníkovi je třeba vědět pouze několik informací. Základem jsou atributy, které není třeba nijak více popisovat, *Jméno* a *Příjmení*. Další důležitou informací o zákazníkovi je jeho adresa, ta se v databázi skládá ze čtyř atributů, těmi jsou *Ulice*, *Město*, *Cp* neboli číslo popisné. Číslo popisné se může skládat jak z čísel, tak i jiných znaků, pro příklad číslo popisné "12/1933a". Posledním atributem tvořícím adresu je *PSČ*. Všechny doposud zmíněné atributy jsou povinné, následující atributy jsou již nepovinné.

Další kategorie atributů by se dala nazvat kontakty. Spadají tam atributy *Email* a *Telefon*, ten je možno uložit i ve formátu s mezinárodní předvolbou (+420). Následující kategorií atributů jsou atributy týkající se firmy. Základním atributem je *Firma*, ten slouží k uložení názvu firmy. Dalším atributem vztahujícím se k firmě je *IČ*. Jedná se o identifikační číslo subjektu nebo organizace, které lze najít na výpisu z živnostenského rejstříku. Následuje atribut *DIČ*. Jedná se o daňové identifikační číslo, které je firmě přiděleno při registraci na finančním úřadu. Posledním atributem této kategorie je *Rejstřík*, zde je možno uvést informaci, ve kterém městě byla firma zapsána do rejstříku na finančním úřadu

Dále jsou použity atributy *Číslo účtu* a *Kód banky*. Posledním atributem je *Smazáno* ten udává informaci o tom, zda již nebyl záznam odstraněn. Záznamy z tabulky nejsou mazány z důvodu, kdyby byl odstraněn zákazník, který je propojen se záznamem faktury. Došlo by tak k nekonzistenci dat a na již vytvořených fakturách by následně chyběly údaje o zákazníkovi. Již z principu je tedy tento atribut také povinný.

Následuje velmi podobná tabulka jménem *Uživatel*, ta obsahuje informace o každém registrovaném uživateli. Proto je spousta atributů totožných s tabulkou *Zákazník*. Jmenovitě se jedná o atributy *Jméno, Příjmení, Ulice, Cp, Metso, Psc, Telefon, Ic, Dic, Firma, Číslo účtu, Kód banky, Rejstřík* a *Smazáno*. Tabulka také obsahuje další specifické atributy, které jsou vázány právě k uživateli. Atribut *Email* je v tomto případě vyžadován, protože je jím podmíněna registrace. Jedním z přidaných atributů je atribut *Aktivní*, který určuje, zda již byl uživatelský účet aktivován. Účet není možné bez aktivace využívat. S tímto atributem se pojí také atribut *Aktivační kód*, ten slouží k uložení automaticky generovaného textového řetězce, který je zaslán uživateli po registraci. Posledním použitým atributem je *SessionToken*, ten je využíván k uchování unikátního kódu (tokenu), který se vygeneruje po přihlášení.

Nejdůležitější tabulkou fakturačního systému je *Faktura*. Téměř všechny její atributy jsou povinné. Prvním z atributů je atribut *Číslo faktury*, ten reprezentuje unikátní označení dané faktury. Atribut *Datum Vystavení*, jak již z názvu vyplívá, reprezentuje datum, kdy byla faktura vytvořena a *Datum splatnosti*, které určuje datum, do kdy musí být faktura uhrazena. Atribut *Zaplaceno* slouží, jak k identifikaci dne, kdy byla faktura zaplacena, tak k určení, zda byla zaplacena (nabývá hodnoty NULL v případě, že faktura nebyla zaplacena). Atribut *Zdanitelné plnění* označuje den, kdy vstupuje v platnost takzvané zdanitelné plnění. To zjednodušeně znamená den, kdy bylo zboží dodáno zákazníkovi, nebo kdy byl vystaven doklad o provedení určité služby zákazníkovi.

Dalším atributem v tabulce *Faktura* je *Stav*. Ten označuje to, v jakém stavu se aktuálně faktura nachází, například "Zaplaceno". Atribut *Stav* je v mírném rozporu s tím, co je uvedeno v předchozích částech textu. Konkrétně se jedná o to, že by atributy reprezentující hodnoty, které jsou předem dány,

měly být zpracovány formou tabulky, kde budou nadefinovány všechny povolené stavy faktury. Ovšem pro zajímavost zůstal tento atribut v databázi v tomto stavu. To umožňuje srovnání rozdílů mezi těmito řešeními v jedné z následujících kapitol. Následují již pouze dva nepovinné atributy a těmi jsou *Vlastní text*, a *Patička.* Atribut *Vlastní text* slouží k definování vlastního textu, který se zobrazí před tabulkou položek v dané faktuře. Druhý atribut *Patička* je pro definici textu, který se zobrazí až za tabulkou s položkami. Všechny ostatní informace pro fakturu jsou řešeny pomocí relací, jak již bylo zmíněno dříve.

S tabulkou *Faktura* je úzce spjata tabulka *Položka*, vzhledem k tomu, že každá faktura musí obsahovat minimálně jednu položku, jinak by postrádala smysl. Základním atributem této tabulky je *Název*. Dále následují atributy *Množství* a *Jednotka*, které také není třeba dále popisovat. Nutným atributem pro každou položku je také *Cena*, konkrétně se jedná o cenu za jednotku. A posledním atributem je *Smazáno*, ten zde plní stejnou funkci jako v ostatních tabulkách.

K položce má vazbu pouze tabulka DPH, která obsahuje pouze jeden atribut a tím je *Hodnota*. Význam této tabulky je pouze definice možných hodnot DPH, které si webová aplikace načítá. Stejným způsobem je vytvořená také tabulka *Forma úhrady*. Jejím atributem je Typ. Další podobnou tabulkou je tabulka *Styl*. Ta obsahuje atributy dva a to *Název* a *Popis*. Poslední tabulkou, jejíž atributy ještě nejsou popsány, je *Vlastní pole*. Ta obsahuje atributy *Název*, *Text* a atribut *Smazáno*.

Tyto tabulky mají tedy přesně definované možné hodnoty, které mohou nabývat. Konkrétně se jedná o tyto hodnoty:

- **DPH**: 0, 10, 15, 21
- **Forma úhrady**: "Hotově", "Bankovní převod", "Platba kartou"
- **Styl**: "Základní", "Moderní"

3.3 Implementace webové služby

Webová služba je napsaná v programovacím jazyce C#, za využití Web API frameworku. Jako taková je rozdělena na část, která obstarává komunikaci s databází a na část, která se stará o naslouchání HTTP dotazům. Služba je postavena na architektuře MVC. Samotná služba je tvořena částmi Model a Controller.

Jako první bude popsána část Controller, která je třídou schopnou zpracovávat HTTP dotazy. Když je tato třída přidána do projektu, jsou v ní automaticky vygenerovány funkce, které obstarávají běžné HTTP metody. Konkrétně se jedná o metody GET, ta se hodí především pro načítání informací. Parametry pro metodu GET se nachází přímo v URI při odesílání dotazu. Druhou metodou je metoda POST, ta je naopak vhodná pro úpravu již existujících dat. Třetí metodou je metoda PUT, ta je vhodná pro přidávání nových záznamů. A poslední vygenerovanou metodou je metoda DELETE. Ta se běžně využívá v případě, že chceme smazat nějaké data.

Nic však nebrání tomu použít každou z těchto metod v controlleru pro jiný účel, než je od ní původně očekáván. Jediným výsledkem definice této metody v controlleru je dosaženo to, že je zavolána příslušná funkce controlleru, které jsou předány parametry dotazu. Pokud nebude některá funkce pro HTTP metodu definována, tak na ni bude webový server vůbec reagovat hláškou, že metoda není na serveru podporována.

Každý controller objekt dědí ze třídy ApiController, kterou nalezneme v namespace System.Web.Http. To umožňuje pracovat s jejími properties jako jsou Request a Url, ty se dají využít například na zjištění původu dotazu.

Pro využití některé z funkcí controlleru stačí pouze ve webové aplikaci vytvořit HTTP dotaz, který využije správnou metodu, bude obsahovat parametry v takovém formátu, jako je definováno v metodě controlleru a použije správnou URL. Pro příklad uvedu URL pro metodu GET controlleru určeného pro přihlášení: "api/Login/<email>". Této cestě musí také předcházet cesta k serveru, takže výsledná URL bude vypadat nějak takto: "http://www.server.domena/api/Login/e@mail.cz". V případě, že server obdrží neplatný dotaz, znamená to, že nenaslouchá požadované metodě, nebo se neshoduje formát parametrů. Server ohlásí, že metoda není na serveru podporována.

Pro implementaci tohoto konkrétního fakturačního systému bylo vytvořeno několik controllerů, každý z nich zprostředkovává komunikaci s jinou tabulkou nebo skupinou tabulek. Text bude dále pokračovat rozborem jednotlivých použitých controllerů a jejich významem.

První controller se jmenuje RegisterController. Ten jak je již z názvu patrné, slouží k registraci. Zprostředkovává tedy komunikaci s tabulkou *Uživatel*. Stará se pouze o přidání nového uživatele do data965báze. Pro tuto funkci má controller implementovanou funkci pro HTTP POST. Součástí controlleru RegisterController je také funkce, která ověřuje, zda email, který byl zadán novým uživatelem, není již registrován. V případě, že ověření emailu proběhne v pořádku, provede se příkaz INSERT do databáze s údaji, které uživatel vyplnil při registraci.

Druhým použitým controllerem je LoginController. Ten je vytvořen za účelem zajištění přihlášení uživatele do systému. Má implementovanou funkci pro obsluhu metody HTTP POST. Ta v případě, že uživatel vyplnil všechny údaje správně, vygeneruje *sessionToken*, který uloží do databáze. V případě, že se přihlašuje uživatel, který ještě nemá aktivovaný účet, je tato funkce ukončena výjimkou a uživatel je o této skutečnosti informován pomocí webového rozhraní. S tímto souvisí druhá funkce implementovaná v tomto controlleru, která se stará o opětovné odeslání aktivačního emailu. K tomuto je uživatel vyzván pouze, pokud má neaktivní účet. Zároveň je pro odeslání emailu nutné specifikovat v parametrech funkce jak email, tak i heslo. Toto je bezpečnostním opatřením, které předchází situaci, že by mohl narušitel pomocí jednoduchého skriptu odesílat aktivační email stále dokola v případě, že by znal neaktivní emaily.

Třetí použitý controller je jednoduchý ActivationController, sloužící pouze k aktivaci účtu nově registrovaného uživatele. Ten má implementovanou funkci pouze pro metodu HTTP POST. Ta za pomocí parametrů *Email* a *Aktivační klíč*, provede aktivaci uživatele v databázi. To znamená, že v tabulce *Uživatel* smaže hodnotu atributu *Aktivační klíč* a přepne atribut *Aktivní* do stavu "true".

Dalším controllerem je UzivatelController. Ten slouží primárně k načítání dat z databáze. Načítá kromě všech atributů uživatele i veškeré informace o fakturách a zákaznících přihlášeného uživatele. Pro tuto funkcionalitu naslouchá metodě HTTP GET. Data jsou po načtení vložena do objektu třídy Uživatel, který je následně vrácen jako odpověď na dotaz GET. Druhá funkce, která je obsažena v tomto controlleru, slouží k aktualizaci údajů uživatele nebo ke změně hesla. Rozdíl je pouze v předaném parametru. Z důvodu bezpečnosti je vždy vyžadován také SessionToken.

Controller, který se stará o tabulku *Zákazník* se jmenuje ZakaznikController. Ten implementuje pouze jednu funkci reagující na HTTP metodu a tou je POST. Ta slouží buď k vložení nového zákazníka, nebo k jeho aktualizaci. Toto rozhodnutí probíhá na základě předaného parametru.

Poslední použitý controller se jmenuje FakturaController. Ten má za úkol postarat se o vkládání nových a aktualizaci stávajících faktur pomocí metody HTTP POST. Rozhodnutí, zda bude faktura vložena či upravena probíhá na základě toho, zda má faktura přiřazené ID.

Další součástí webové služby je Model. Ten je reprezentován třídami, kde každá z nich přesně zrcadlí jednu tabulku z databáze. Třída se tedy jmenuje vždy stejně jako její ekvivalent v databázi a všechny její vlastnosti (proměnné) odpovídají atributům tabulky. Každá proměnná má datový typ, který je shodný s datovým typem použitým v databázi. Poslední součástí modelu jsou konstanty, které určují velikost jednotlivých proměnných typu string (VARCHAR v databázi). Každá proměnná tabulky v databázi musí mít definovanou velikost, zatímco v C# toto omezení neplatí. Je nutné si jej tedy vytvořit uměle za pomocí konstant.

Pro komunikaci s databází slouží třída jménem *Database*. Pro její správnou funkčnost je nutné použit namespace System.Data.SqlClient. Pro připojení k databázi je třeba vytvořit tzv. "SQL connection string", což je textový řetězec obsahující adresu serveru, název databáze, přihlašovací jméno a heslo ve standardně definovaném tvaru. Connection string může obsahovat i další parametry, jako je například timeout, díky kterému bude pokus o připojení k databázi ukončen po přesně stanoveném čase.

Třída *Database* obsahuje funkce, které se starají o vytváření a ukončování spojení s databází. Dále je možné pomocí ní odesílat dotazy a příkazy do databáze. Jsou zde implementovány také funkce pro práci s transakcemi. Transakce jsou důležité pro udržení konzistence dat. V případě, že například selže spojení během provádění zápisu do databáze, provede se pouze rollback, který vrátí všechny změny do původního stavu. V případě, že by nebyly použity transakce, by se mohlo stát, že změna bude provedena ve webové aplikaci, ale v databázi ne.

Třída *UzivatelTable* je určena k tomu, aby vytvářela konkrétní databázové příkazy týkající se především tabulky Uživatel. Tato třída je statická a je využívána controllery. Třída obsahuje funkci pro vložení (registraci) uživatele do databáze. Tato funkce nejdříve ověří, zda již uživatel není registrován (jeho email není v databázi) a vygeneruje nové ID uživatele. Toho dosáhne načtením posledního ID z databáze pomocí jednoduchého SQL dotazu. Následně uživateli vygeneruje aktivační klíč a vytvoří pomocí SQL příkazu INSERT nový záznam v databázi. Poslední částí této funkce je odeslání emailu uživateli, ve kterém je informován o proběhlé registraci. Email obsahuje odkaz pro aktivaci, který je po otevření zpracován webovou aplikací. Toto vede k zavolání funkce sloužící k aktivaci. Ta v první fázi zkontroluje, zda již účet nebyl v minulosti aktivován. Pokud ano, vyvolá výjimku, kde o této skutečnosti informuje uživatele. Ve druhé fázi aktualizuje záznam v databázi tak, že změní hodnotu atributu *Aktivní* na *true* a vymaže aktivační klíč.

Další implementované funkce slouží pro úpravu uživatele a aktualizaci hesla. Ty aktualizují záznam v databázi podle předaných parametrů pomocí SQL příkazu UPDATE. Vždy je potřeba aby funkce dostala platný SesionToken, v opačném případě nebude změna povolena a bude vyvolána výjimka. Další implementovaná funkce této třídy slouží ke znovu odeslání aktivačního emailu, to je potřeba v případě, že si o to uživatel požádá ve webové aplikaci.

Jedna z rozsáhlejších funkcí slouží k načtení všech údajů vztahujícím se k uživateli. Ta má za úkol stáhnout z databáze jeho údaje, zákazníky a faktury. Rozsáhlost této funkce spočívá v tom, že volá několik více specifických funkcí pro načítání dat. První z funkcí načte informace o aktuálně přihlášeném uživateli. Další funkce načte seznam všech zákazníků, kteří patří k přihlášenému uživateli. Ten se následně uloží do objektu třídy Uživatel. Další informace, které je třeba načíst, jsou faktury a její položky. Funkce pro stažení těchto informací opět pomocí jednoduchého SQL SELECT dotazu stáhnou veškeré potřebné data o každé faktuře a jejich položkách. Tento seznam uloží do objektu Uživatel. Výsledkem provedení všech těchto funkcí je pak kompletní objekt Uživatel, obsahující všechny data, které budou následně předány webové aplikaci.

Poslední zajímavou funkcí v této třídě je funkce pro přihlášení uživatele. První činnost, kterou funkce vykoná je ověření, zda je uživatelův účet aktivní za pomocí krátkého SQL dotazu. V případě, že účet je aktivní, bude vygenerován *SessionToken*, který bude předán webové aplikaci. Pokud však účet aktivní není, bude vyvolána výjimka, která popisuje, k jaké chybě došlo. K tomuto účelu byla vytvořena nová podmínka jménem NotActivatedException, aby bylo jednoduché rozeznat podmínku poukazující na to, že uživatel není registrován. Následně, po vygenerování SessionTokenu, je odeslán do databáze. Tento atribut je důležitý z důvodu ověřování, zda je uživatel již přihlášen. Pokud aktualizace databáze proběhne v pořádku je SessionToken předán webové aplikaci, která rozhodne o dalším postupu.

Poslední použitou třídou v rámci webové služby je třída *TablesOperations*, která obsahuje funkce pro načítání a editaci jiných tabulek než tabulky *Uživatel*. Funkce, které zpracovávají SQL UPDATE i SQL INSTERT jsou velmi podobné funkcím UzivatelTable, není nutné je tedy dále rozebírat.

4 Návrh a implementace webové aplikace

Tato kapitola upřesní význam webové aplikace v rámci celého bakalářského projektu. Zároveň popíše to, jak spolu jednotlivé součásti webové aplikace spolupracují. Bude představen vzhled webové aplikace a všechny funkce, které jsou nabídnuty uživatelům. Zároveň zde bude popsáno, jak byly jednotlivé funkce navrženy a následně naimplementovány.

4.1 Význam webové aplikace a její návrh

Webová aplikace je tou nejdůležitější součástí vytvořeného projektu. Je to totiž jediná část celého projektu, se kterou budou pracovat všichni uživatelé. Proto je nutné, aby v této aplikaci byly veškeré ovládací prvky, které může uživatel potřebovat. Žádný jiný způsob přístupu k uloženým datům než za pomocí této aplikace není uživatelům umožněn. Vývojář má samozřejmě přístup přímo k samotné databázi a je tedy schopen ji spravovat i bez využití webové aplikace.

Aplikace je navržena tak, aby mohla být umístěna na serveru, který je přístupný skrze Internet. Vzhledem k tomu, že tento projekt slouží pouze jako prezentace bakalářské práce a není v plánu jej dále využívat ke komerčním účelům, není webová aplikace umístěna na žádný veřejně dostupný webový server. Jedním z důvodů pro toto rozhodnutí je aktuálně velká konkurence, která má k dispozici mnohem více zdrojů a zároveň jsou její aplikace na mnohem vyšší úrovni. Nebylo by tedy vůbec jednoduché získat si své zákazníky, kteří by měli zájem využívat tohoto fakturačního systému. Pro prezentaci je webová aplikace vždy spuštěna na lokálním serveru v rámci vývojářského nástroje Microsoft Visual Studio, konkrétně je aplikace spuštěna na serveru IIS Express, který je součástí Visual Studia.

Samotná webová aplikace je rozdělená na dvě logické části. První částí je prezentační, ta je přístupná komukoliv a kdykoliv. Je to jednoduchá HTML stránka, která jednoduše popisuje, k čemu slouží tento fakturační systém, jaké jsou jeho výhody nebo například ceník služeb. Další z možností, které nabízí tato stránka je registrace a přihlášení do systému. Druhou částí webové aplikace je takzvaný Dashboard. To je část aplikace přístupná pouze přihlášeným uživatelům. Tady může uživatel pracovat se svými fakturami. Může je zde vytvářet, upravovat i mazat. Dále zde může spravovat svůj vlastní profil nebo spravovat záznamy v adresáři svých zákazníků. Je zde také možné zobrazit si přehledné statistiky o svých fakturách.

Pro implementaci chování webové aplikace byl podle zadání zvolen programovací jazyk TypeScript, který je již součástí vývojářského nástroje Microsoft Visual Studio 2015. Ten obsahuje podporu automatického našeptávače IntelliSense, a také kontrolu chyb již v době psaní samotného kódu. Kód je následně automaticky přeložen do programovacího jazyka JavaScript. Tyto vlastnosti z něj dělají velmi příjemným programovacím jazykem vhodným pro naprogramování chování webové aplikace. Rozložení a veškeré součásti webové aplikace jsou napsány pomocí značkovacího jazyka HTML5. Ten je v celém svém rozsahu automaticky generován za pomocí JavaScriptu v době běhu aplikace. Samotný HTML soubor stačí pro vytvoření takové webové stránky, která bude v pořádku fungovat. Avšak její vzhled nebude přehledný a neumožňuje dostatečné možnosti rozmístění jednotlivých elementů na stránce. Proto jsou současně využity kaskádové styly CSS. Za jejich pomocí je velmi snadné vytvořit jakékoliv stránce požadovaný vzhled. Umožňuje pracovat s velikostmi, umístěním, barvami nebo vytvářením animací. Spojením všech vyjmenovaných součástí vznikne celistvá, přehledná a funkční webová aplikace.

4.2 Implementace webové aplikace

Základ webové aplikace tvoří jediný staticky nadefinovaný HTML soubor, který se jmenuje *"index.html"*. Je to stránka, která je vždy načtena jako první. Slouží pouze k definici základní kostry stránky a nastavení odkazu na použité skripty, styly nebo knihovny. To znamená, že obsahuje pouze prvky head a prázdné body. V prvku head jsou definovány veškeré použité odkazy a title stránky. V případě, že je nutné přidat například nový JavaScript soubor do projektu, je nutné jej přidat také do tohoto HTML souboru.

Prvním skriptem, který je spuštěn okamžitě po načtení stránky je *"GUI.js"*. Tento skript je implementován za účelem vytvoření úvodní stránky webové aplikace. V tomto skriptu je vytvořena statická třída HashDirector, která se stará o práci s hashem. Hash je část URL, která je vždy na konci a je oddělena znakem "#". Pro příklad je vyznačen tučně hash v následující URL http://localhost:54516/**#Login**. Veškeré přechody mezi stránkami jsou řešeny právě změnou tohoto parametru. Toto řešení je výhodné v tom, že i přes využití dynamicky vytvořené stránky, je stále funkční tlačítko zpět. V případě, že by nebyl hash použit, byl by uživatel po kliknutí na tlačítko zpět v prohlížeči odkázán na stránku, ze které přišel. Nevýhodou využití dynamicky generované stránky je ten, že je nutné mít povolený v prohlížeči JavaScript. Ovšem bez něj by tato aplikace fungovat nemohla, takže v tomto projektu není nutné brát tuto nevýhodu v úvahu.

Třída HashDirector po spuštění metodou *start* funguje tak, že naslouchá události objektu window, která je vyvolána po změně hashe. Druhou velmi důležitou funkcí, která je v této třídě implementována, je funkce *Parse*. Ta se stará o správné přečetní hashe z URL a podle výsledku buďto vykreslí úvodní stránku nebo předá nutné informace statické třídě *DashBoard*, která se již postará o další zpracování této informace. Během parsování tato funkce musí přečíst cookie uložené u uživatele, ten totiž může obsahovat SessionToken společně s emailem. To by značilo, že je již uživatel přihlášen a bude mu umožněno přejít rovnou do Dashboardu. Tato situace může nastat v případě, že uživatel se přihlásí do systému, ten následně po pár minutách ukončí, ale neodhlásí se. V tomto případě zůstává uživatel přihlášen, než bude cookie automaticky po určitém čase vymazáno.

Pro vykreslení úvodní stránky je v tomto skriptu vytvořena statická třída s názvem *Layout*, která podle aktuálního hashe zavolá funkci DrawGUI příslušné třídy. Každá třída zde reprezentuje jednu stránku a její funkce DrawGUI slouží pro vykreslení všech prvků na stránku. Jinak řečeno tato funkce slouží pro správné vygenerování HTML souboru.

Každá HTML stránka se skládá z mnoha prvků (elementů). Ty mají vždy stejný syntax a to například * * pro vytvoření prvku div a * * pro jeho ukončení. K tomu je v TypeScriptu implementován typ *HtmlElement*. Ten je však pro účely tohoto projektu nedostatečný. Proto byla v tomto projektu vytvořena třída s názvem *LayoutEement*. Třída obsahuje nejen samotný *HtmlElement*, ale za pomocí konstruktoru umožňuje vytvoření jakéhokoliv HTML prvku. Prvek má nastavenou dostupnost na protected, proto je v této třídě funkce pro získání HtmlElementu a funkce, která umožňuje jednoduše nastavit animaci po najetí myši na prvek.

Poslední třídou, která je součástí tohoto skriptu je třída *TopMenu*. Ta slouží k vytvoření celé základní nabídky na úvodní stránce. Toto menu slouží k navigaci na stránce. Mezi položky, které je možné vybrat patří položky: "Domů", "Info", "Galerie", "Cena" a "Login". Tato třida dědí z třídy *LayoutElement*, což znamená, že je konkrétním prvkem na stránce.

Po kliknutí na kteroukoliv položku menu se změní hash stránky. To znamená, že z pohledu prohlížeče se uživatel přesunul na jinou stránku, ačkoliv webová aplikace má pouze jedinou stránku a ta je pouze dynamicky upravována.

Dalším skriptem v projektu je *HomePage*, který obsahuje pouze jedinou třídu stejného názvu a funkci DrawGUI na vykreslení úvodní stránky. Ta se vykreslí stejně při kliknutí na jakoukoliv položku menu vyjma položky *Login*. Jediným rozdílem je pozice obrazovky na této stránce, na kterou bude uživatel po kliknutí odkázán. Celá hlavní stránka totiž obsahuje všechny informace o webové aplikaci přímo pod sebou.

Po kliknutí na položku menu s názvem *Login* bude zavolána funkce pro vykreslení stránky pro přihlášení uživatele. Ta je složená z jednoduchého formuláře, kde se vyplňuje přihlašovací jméno a heslo. Mezi další možnosti patří volba, zda si uživatel přeje ukládat data do počítače v případě, že by na něm chtěl v budoucnu pracovat v režimu offline. Dále se zde vyskytují dvě tlačítka - jedno pro registraci, které přesune uživatele na formulář určený k registraci do systému a druhé pro přihlášení. Z funkčního hlediska funguje tento skript tak, že se vytvoří objekt z třídy *LoginElement*, která je rozšířením třídy *LayoutElement*, což umožňuje jednoduše vytvořit nový prvek na stránce. Zbývající část této funkce pouze vykreslí formulář, který byl speciálně pro tento projekt navržen za pomocí vlastní třídy, které bude dále popsána. Pro přihlášení je za pomocí funkce knihovny jQuery vytvořen AJAX, což je technologie, která slouží k vytváření HTTP dotazů. Těm v tomto projektu naslouchají dříve naimplementované controllery popsané v předchozí kapitole. AJAX dále určuje, co se má provést v případě úspěchu či neúspěchu dotazu. Pro funkci přihlášení tedy v případě úspěchu přihlásí uživatele do systému a v opačném případě vypíše chybu, která nastala při vykonávání dotazu. Za úspěšný dotaz se pro AJAX počítá také neúspěšné přihlášení. Rozhodnutí zda byl uživatel přihlášen, či nikoliv je tedy provedeno podle vrácené hodnoty ze serveru.

Dále budou popsány třídy, ze kterých se skládají formuláře. Účelem těchto tříd je zjednodušení a zkrácení zápisu při vytváření formuláře. Zároveň jim také přidává další funkce. Název základní třídy je *ValidableForm*, již z názvu vyplývá, že součástí této třídy je také validace vstupů. Pro vytvoření formuláře stačí pouze napsat jeho název při vytváření objektu a formulář bude připraven. Pro formulář je důležité, aby měl nějaké vstupy. Ty se vytvářejí jako objekty ze třídy *FormInput*. Vstupu se pro vytvoření předají potřebné parametry a pomocí jediného řádku je vstup přidán do formuláře. Mezi parametry je třeba určit především název vstupu, jeho typ, pro který je vytvořen enum a zvolit zda bude nutné tento vstup vyplnit nebo může zůstat prázdný. U každého vstupu je možno také ověřit, zda neobsahuje neplatné hodnoty. Pro tento účel je využita funkce, která za pomocí regulérních výrazů rozhodne, zda je vstup platný či nikoliv. Ten je možno také nastavit jako parametr při vytváření objektu této třídy. Konstruktor této třídy tedy vytvoří nový HTML element, který následně předá do formuláře.

Funkce, která ve formuláři testuje, zda je celý formulář platný, pouze projde všechny nadefinované vstupy a ty jednotlivě otestuje. Poslední součástí této třídy je funkce pro vypsání informačních a chybových hlášek. Ty se vždy zobrazí pod formulářem. Za pomocí této třídy je tedy dosaženo toho, že je velmi jednoduché vytvořit nový formulář, který je následně umístěn do stránky. Je také velice jednoduché zjistit, zda jsou všechny vstupy platné a díky použití regulérních výrazů není nutné vytvářet speciální funkce pro každý formulář, které by zpracovávaly neplatné vstupy. Je tím také zajištěno, že budou mít formuláře jednotný vzhled a funkčnost.

Poslední stránkou, která je součástí úvodní stránky, je Registrace. Ta je prováděna ve dvou fázích. V první uživatel vyplní uživatelské jméno a heslo, případně může také vyplnit IČ, pokud jej má. Po potvrzení bude vytvořen AJAX, který ověří, zda zadaný uživatel nebyl již v minulosti vytvořen a zároveň ověří, zda heslo splňuje veškeré požadované podmínky pro jeho vytvoření. Vzhledem k tomu, že se nejedná o projekt, který by měl být využíván komerčně, je jediným omezením minimální počet znaků hesla na 8 znaků. V opačném případě by muselo heslo obsahovat ideálně číslici, velké písmeno a jeden speciální znak. Pokud proběhne ověření první fáze registrace v pořádku, zobrazí se zbytek formuláře. Ve zbytku formuláře je třeba vyplnit základní údaje o uživateli, jako jsou jméno, příjmení a celá adresa. Poslední povinnou položkou je opakované vepsání hesla a registrace může proběhnout. Další položky formuláře je možno doplnit, ale povinné nejsou. Jedná se o údaje o firmě, bankovním účtu a kontaktním telefonu.

Po úspěšné registraci je uživateli zaslán aktivační email. K aktivaci slouží jednoduchá stránka označená hashem #Aktivace. Ta slouží k informování uživatele o průběhu aktivace, a umožňuje mu požádat o opětovné odeslání aktivačního emailu.

Po úspěšné aktivaci je uživatel přihlášen do systému. Pro ten je v projektu vytvořena celá složka pojmenovaná Dashboard, jež obsahuje všechny použité skripty, které se týkají pouze druhé části webové aplikace. Toto oddělení slouží především pro přehlednost. Tato stránka má některé třídy podobné s třídami použité pro práci s úvodní stránkou, avšak žádnou z nich nesdílí a jsou vytvořeny zvlášť, většinou z důvodu, že jsou zde požadovány jiné vlastnosti a funkce elementů než na úvodní stránce.

Základním skriptem, který je použit pro vytvoření dashboardu je skript nazvaný MainPage, který slouží jako startovní bod pro veškerou práci v dashboardu. Základní třídou, na které je dashboard postaven nese stejný název, také se tedy jmenuje DashBoard. Jedná se o statickou třídu, postupně postaví celou stránku a odkazuje se na další skripty sloužící k práci s ostatními částmi dashboardu. Tato třída si také uchovává informaci o přihlášeném uživateli, to znamená všechny jeho údaje a data, které vložil do fakturačního systému. Tyto informace musí však nejprve načíst z databáze. Pro tento účel je využit AJAX, který za pomocí HTTP GET dotazu stáhne všechny potřebné údaje.

Pro umožnění typové kontroly objektů stažených z databáze a následnou prací s nimi je v projektu vytvořen jeden skript, který simuluje modelovou část celého projekt, ale pouze pro účely TypeScriptu. To znamená, že jsou v tomto skriptu vytvořeny třídy pro každou tabulku z databáze. Oproti modelu, napsaném v programovacím jazyce C#, tedy na straně serveru obsahují tyto třídy také konstruktory. Konstruktory se následně využívají pro vytváření instancí z těchto tříd, například při načítaní dat z databáze. Některé z tříd obsahují také funkce, které jsou následně využívány různými skripty za běhu aplikace. Tento skript také obsahuje pro některé třídy enumy, které se vztahují k některým třídám.

Pro příklad některé metody implementované v těchto třídách, je metoda na kopírování objektu. Metoda funguje tak, že překopíruje hodnoty z jednoho objektu do druhého. Oba musí být instancemi stejné třídy. Další funkce jsou například pro načítání seznamu zákazníků nebo načtení jména uživatele v předem zadaném tvaru. Třída faktura má implementovanou funkci pro výpočet její celkové ceny, jelikož tato cena se do databáze neukládá. Pro definici stavu faktury je použit enum, který obsahuje řetězcové hodnoty jako "Nová" nebo "Zaplaceno". Podobné funkce se pojí také k třídě *Položka*. Ty slouží pro výpočet ceny podle toho, jaká hodnota byla zadána do formuláře. Tyto funkce budou zmíněny v další části textu.

Jakmile jsou všechna data načtena, může být vykreslen dashboard. Ten je vykreslován postupně a skládá se ze tří částí. První částí je horní panel, který obsahuje název systému, jméno uživatele nebo firmy a adresu. Hned vedle se nachází ikonka uživatele, která po rozkliknutí zobrazí stránku editace profilu uživatele. Posledním tlačítkem v horním panelu je odhlášení. To uživatele odhlásí, vymaže jeho SessionToken z prohlížeče a přesměruje jej na úvodní stránku webové aplikace.

Druhou částí je postranní nabídka, která slouží k pohybu mezi stránkami dashboardu. Pro tuto nabídku je vytvořena třída jménem *HtmlMenu*. Její účel je zkrátit a zjednodušit zápis pro vytvoření postranní nabídky. Z pohledu HTML je nabídka vytvořena pomocí elementu *ul*, neboli nečíslovaného seznamu. K vytvoření nové položky nabídky stačí pouze zavolat funkci *AddMenuItem*, které se předají parametry jméno, id, podle kterého lze prvek v seznamu najít, objekt stránky na kterou bude tento prvek odkazovat a ikona. Ikona je zde vytvořena za pomocí knihovny Font Awesome. Ta poskytne vývojáři sadu vektorových ikon, které lze upravovat za pomocí kaskádových stylů stejně jako jakýkoliv text. Následuje příklad kódu pro vytvoření položky v menu.

```
menu.AddMenuItem("Profil", "Profil", new Page(user), "fa-user");
```

Třetí částí grafického rozhraní je takzvaný wrapper. To je samotný obsah stránky vymezený pomocí horního menu a postranní nabídky. Při pohybu mezi stránkami dashboardu se tedy znovu překreslí pouze tento wrapper a vše ostatní zůstává nezměněno. Wrapper je v tomto projektu definován také za pomocí třídy a z něj vychází celková struktura každé stránky. Samotný wrapper je jednoduchý div element. Ovšem tato třída si udržuje také hlavičku wrapperu a jednotlivé sekce. Každá z těchto součástí je také definovaná vlastní třídou. Hlavička wrapperu obsahuje text, neboli název celé stránky a může obsahovat tlačítka. Pro správné vykreslení byla pro tuto třídu implementována funkce Draw, která celou hlavičku vykreslí.

Sekce jsou další součástí wrapperu. Každá sekce má svůj název, a může obsahovat tlačítka, formuláře nebo jakýkoliv vlastní HTML element. Tlačítka se zobrazují vždy na konci každé sekce, její název zase v hlavičce sekce. Tato třída má také funkci Draw pro správné vykreslení sekce, takže vývojář pouze naplní tuto sekci požadovanými prvky a za pomocí této funkce bude výsledná sekce vypadat vždy stejně.

Jednou z velmi důležitých tříd, která se váže na wrapper je *WrapperForm*. Jedná se o třídu reprezentující formulář, který je uzpůsobený tomu být umístěn do wrapperu. Důvodem k jeho vytvoření bylo zachování přehlednosti a jednoduchosti, které je docíleno právě tímto řešením. K přehlednosti také velice pomohlo to, že je webová aplikace napsána v programovacím jazyce TypeScript. Umožnila totiž vytváření tříd s podporou typové kontroly. Formulář je vytvořen pomocí HTML prvku form. Avšak tento formulář nemá položky tvořené pouze obyčejným vstupem, ale prvkem, kde je ke vstupu přilepen i label a celý prvek je následně vložen do formuláře. Formulář tedy musí obsluhovat funkci pro přidání takového pole, a tato funkce by neměla být složitá pro zavolání, protože je třeba ji při vytváření jednotlivých stránek volat mnohokrát.

Pro toto pole je vytvořena třída *FormField*. Její konstruktor je jednoduchý a obsahuje i několik nepovinných parametrů. Základem jsou parametry jméno, typ a id. Typy jsou použity stejně, jako je definován pro HTML5 prvek input. Mez nepovinné atributy se řadí hodnota pole, informace jestli je nutné tento vstup vyplnit, jestli je aktivní a uzamčený. Tento konstruktor tedy nabízí hodně možností, jak vytvořit vstup pro formulář.

Základem je HTML prvek input. Tomu je přiřazeno ID a typ. V případě, že je nadefinován některý z nepovinných parametrů, je také přiřazen výslednému prvku. Pro název tohoto vstupu je pak vytvořen prvek label, který je vložen nad samotný vstup. Jedním z volitelných parametrů byla možnost uzamknout tento prvek, což znamená, že již nebude aktivní, ale styl textu zůstane stejný, jako když je aktivní. To z toho důvodu, že když je prvek neaktivní, je i text ve vstupu méně viditelný a nehodí se to pro všechny případy. V případě, že bude vstup nastaven tak, že musí vyžadovat hodnotu, je před vstup vložena hvězdička za pomocí ikony z knihovny Font Awesome. Poslední věcí, která se provede při vytváření vstupu, je přidání animací. Ty jsou zde využity na posun popisu vstupu nahoru v případě, že vstup obsahuje nějaký text.

Existuje jeden specifický případ vstupu a tím je datalist. Jedná se o textový vstup, který je však možné vybrat z rozevírací nabídky během psaní textu. Tu je nutné nejprve naplnit hodnotami a při změně vstupu je třeba porovnat zadaný text s hodnotami. Pokud by se neshodovaly, nebyl by vstup platný.

Další součástí této třídy je ověření vstupu, to je v tomto případě trochu složitější oproti ověřování formuláře představeném pro úvodní stránku. Pracuje s objektem validity, který je součástí každého HTML prvku input. Ten rozezná konkrétní chybu, která na vstupu nastala. Může se stát například, že uživatel zadá hodnotu, která se neshoduje s typem, který byl pro tento vstup nastaven. Jedná se o typy přímo definované pomocí vlastní třídy. Ta uchovává ke každému definovanému typu regulární výraz, pomocí kterého je jednoznačně určeno, zda je tvar vstupu správný, také definuje zprávu, která se uživateli zobrazí v případě, že vyplní vstup v nesprávném tvaru. Například se může jednat o typ telefon, kde bude ověřeno, zda je vstup do pole telefon zadán ve správném tvaru se znakem "+" a mezinárodní předvolbou nebo bez předvolby. Ostatní tvary zde nejsou povoleny. Další z běžných chyb, které jsou třeba hlídat je, zda jsou vyplněna všechna pole, která jsou nastavena jako povinná, nesprávné rozmezí hodnot vstupu (především u číselných hodnot) aj. Poslední ověření se týká jakékoliv chyby, lze totiž nastavit i ověření chyb, které nebyly přímo definovány. Následně v případě, že nastane jakákoliv chyba, je vytvořena chybová zpráva pod vstupem, kterého se chyba týká.

Třída pro formulář, sekci i celý wrapper pak má funkce pro odemčení a uzamčení všech vstupů, které obsahují. Formulář ještě navíc obsahuje funkci sloužící k validaci všech vstupů v tomto formuláři.

Základní rozhraní dashboardu je tedy vykresleno a funkce pro jeho vytvoření pokračuje parsováním hashe. Hash pro dashborad je vždy ve tvaru *#Dashboard/<stránka>/<podstránka>*. Proto funkce pro parsování vždy kontroluje, jestli hash začíná slovem Dashboard, a jestli jeho délka není příliš dlouhá. Následně projde jednotlivé nadefinované sránky z menu a podle hashe otevře tu, které hash odpovídá. V případě, že by se chtěl do dashboardu takto dostat nepřihlášený uživatel, nebude mu to umožněno a zobrazí se mu pouze přihlašovací obrazovka.

Následují skripty pro jednotlivé stránky dashboardu. Každá stránka má svůj vlastní skript z důvodu logického oddělení tříd a funkcí od jiných stránek. Třída stránky vždy dědí ze třídy DashboardPage. Toto rozšíření umožňuje vytvořit jednoduchý konstruktor s názvem stránky, který vytvoří nadpis stránky a po jeho vytvoření zavolá funkci *DrawGUI* pro vykreslení celé stránky, která se postará o úpravu ikon a vytvoření vnitřního wrapperu. Následně zavolá funkci pro vykreslení vnitřního grafického rozhraní. V ní se již vytváří sekce, ve které se dále vytváří samotný formulář sloužící k zobrazení a úpravě profilu. Formulář je vytvořen tak, že pokud se vejde na obrazovku, bude obsahovat vždy dva sloupce. Tyto sloupce jsou vždy reprezentovány samostatným formulářem. U každého pole je

nutné vyplnit požadované atributy a nakonec zavolat funkci pro vykreslení sekce. Dále se k jednotlivým tlačítkům u formulářů přiřadí funkce.

Jako první bude představen skript a stránka sloužící k úpravě uživatelského profilu. Na této stránce se vyskytují dvě sekce, jedna slouží k úpravě samotného profilu a druhá slouží ke změně hesla. Uživatel může ve svém profilu měnit veškeré osobní informace, a zároveň i informace týkající se jeho firmy, či pouze podnikání. Sekce obsahuje dvě tlačítka, jedno slouží k resetování veškerých hodnot do původního stavu a druhé k odeslání změn do databáze. Odeslání údajů do databáze proběhne tak, že se vytvoří AJAX a jako parametr se mu předá objekt upraveného uživatele. Druhou sekcí na stránce je úprava hesla, zde stačí vyplnit pouze původní a nové heslo. Pokud je vše v pořádku, bude požadavek na změnu hesla odeslán do databáze.

Dalším skriptem, a tedy i stránkou je seznam faktur. Ta slouží pouze k vykreslení tabulky všech faktur, které jsou uloženy v databázi přihlášeným uživatelem. V záhlaví stránky je přidáno tlačítko na přidání nové faktury, kterým bude uživatel odkázán na stránku, kde si může fakturu přidat do databáze. Tabulka není na této stránce zpracována jako HTML tabulka, ale jako prvek div. Ten v sobě obsahuje některé informace z faktury. Patří mezi ně číslo faktury, jméno zákazníka, datum vystavení, celková cena a stav. Na konci každé položky je přidáno tlačítko na rychlé vymazání faktury.

Velmi podobným způsobem je zpracována stránka obsahující seznam zákazníků. Jediným rozdílem je obsah. Tlačítko v záhlaví slouží k vytvoření nového uživatele a v seznamu jsou parametry týkající se uživatele, jako jeho jméno, adresa a email.

Z tohoto seznamu vyplývá nutnost mít stránku, která umožní pracovat s údaji zákazníků. Ta slouží jak k vytváření nového zákazníka, tak k editaci stávajícího zákazníka. V obou případech je vytvořena na stránce sekce, ve které je následně vykreslen formulář. V tomto formuláři jsou téměř totožné položky jako ve formuláři, který slouží pro editaci údajů v profilu uživatele. Jediným polem navíc je zde možnost změnit email zákazníka. Tlačítka v této sekci slouží pro obnovení změn, potvrzení změn a smazání zákazníka. Samotná úprava i smazání proběhnou tak, že se vytvoří AJAX, kterému je do parametru předán objekt upraveného zákazníka. Smazání pouze změní atribut smazáno v tabulce, takže se také jedná pouze o editaci záznamu.

Jednou z nejdůležitějších stránek ve fakturačním systému je FakturaPage. Ta slouží k vytváření a editaci faktur. Tato stránka je vytvořena ve dvou režimech. Prvním z nich je zobrazení faktury. V tomto zobrazení jsou zobrazeny informace o faktuře - její číslo, zákazník, stav faktury, datum vystavení a splatnosti, nakonec také datum zaplacení a vybraný styl faktury pro tisk. Další sekce zobrazuje všechny položky faktury v přehledné tabulce. V této tabulce jsou uvedeny informace, jako je název položky, její množství, jednotka a ceny, jak s DPH, tak bez DPH. V zápatí této tabulky jsou nakonec zobrazeny veškeré součty cen.

V hlavičce stránky jsou následně vykresleny dvě tlačítka, jedno slouží ke smazání faktury a druhé k přechodu do editace faktury. Tlačítkem odstranit se pouze edituje záznam v databázi pro fakturu, konkrétně její atribut Smazáno. Do doby než je zvolena možnost editace, není možné do formuláře vkládat jakékoliv údaje. Po přechodu do editačního režimu se u položek ve formuláři zobrazí, které jsou povinné a které nikoliv. Dále je možné změnit jakoukoliv položkou vyjma položky stav, ta je automaticky upravena společně s údaji ve faktuře. V tomto režimu se také zobrazí možnost editovat položky faktury. To znamená, že místo názvů a cen se zobrazí příslušné vstupy, do kterých je možné psát požadované data. Na konci každého řádku v této tabulce položek je tlačítko na smazání položky.

Přidání položky se provede klikem na tlačítko, které je zobrazeno v záhlaví této tabulky. To pouze přidá řádek do této tabulky a její správné vyplnění se ověřuje ve chvíli, kdy uživatel klikne na tlačítko pro uložení faktury. Veškeré ceny, jak celková, tak části uvedené v položce jsou automaticky vypočítány a dosazeny okamžitě po úpravě jakékoliv ceny. Není tedy nutné nijak dopočítávat cenu, ale stačí vyplnit kteroukoliv položku, a systém si ostatní dopočítá automaticky. Tabulka položek má také svůj vlastní systém ověřování vstupu. Jedná se však pouze o to, zda je vyplněné pole se jménem položky, zbývající hodnoty se po nesprávném vstupu automaticky opraví na minimální hodnotu. Položky s nulovou cenou jsou v tomto systému povoleny. Uložení faktury se provede přes AJAX, kde se odešle změněná faktura, která bude následně změněna v databázi a zároveň automaticky upraví i položky faktury.

Stránka pro statistiky slouží pro zobrazení grafů a tabulek. Jedná se o grafy a tabulky zobrazující počet vystavených faktur za určité období, počet uložených zákazníků nebo celková cena zaplacených a nezaplacených faktur. Tyto grafy jsou vykresleny za pomocí API poskytnutého od společnosti Google. Konkrétně se jedná o Google Charts. Pro vykreslení grafu stačí využít nadefinovaných funkcí tohoto API.

Na přehledové stránce, která je zobrazena jako první po přihlášení uživatele do systému, můžeme nalézt zkrácený seznam faktur i zákazníků a některé grafy. Jeho hlavním úkolem je předložit uživateli veškeré informace co nejrychleji a nejpřehledněji. Využívá stejných funkcí jako skripty jednotlivých stránek.

Součástí tohoto projektu byla také možnost vygenerovat faktury do tisknutelné podoby. Pro tento účel je nejvhodnější soubor typu PDF. Fakturu je nutné vytvořit krok za krokem pomocí funkcí použité knihovny. Pro účely tohoto projektu byla zvolena knihovna jsPDF. Ta je schopna vytvořit PDF soubor vytvořit zcela od začátku. Pro tento účel je v projektu definována třída PDF, té se předá objekt faktury a ta už se postará o generování.

Další stěžejní součástí tohoto projektu bylo umožnění funkčnosti fakturačního systému i bez přístupu k internetu. K tomuto účelu je potřeba splnit dvě podmínky. První podmínkou je, aby nebylo nutné vždy přistupovat k databázi. Na jednu stranu je sice možné, že uživatel nebude pracovat s aktuálními daty, na stranu druhou však nebude omezen výpadkem internetu nebo jen databázového serveru. Druhou podmínkou je to, že webová aplikace musí být dostupná také bez přístupu k internetu. Nemělo by žádný význam umožnit uživateli práci pouze bez přístupu k databázi.

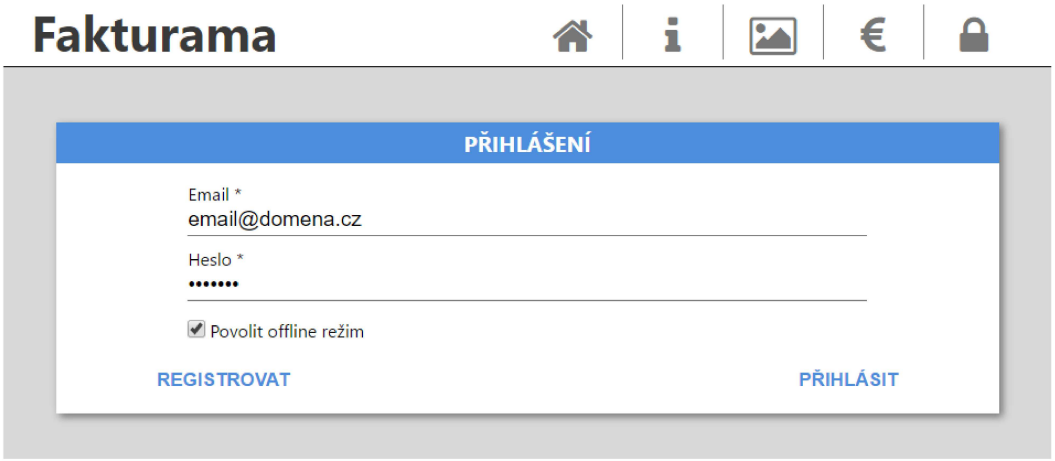
Pro zobrazení webové aplikace v režimu offline je potřeba, aby byla aplikace uložena v počítači. K tomuto se používá cachování. Ke správnému caschování je nutné nadefinovat, které části webové aplikace budou dostupné i offline, ty se následně stáhnou a uloží do prohlížeče uživatele při první návštěvě stránky. Zároveň tato možnost zvýší rychlost načítání stránky, jelikož ji není nutné pokaždé stahovat ze serveru. Stažena je vždy pouze aktuální verze, a tím tato technologie ulehčí zátěž serveru.

V případě, že je uživatel aktuálně bez připojení k internetu, ale potřebuje vytvářet nové faktury či upravovat faktury starší, je potřeba, aby tyto změny byly uloženy na straně uživatele. K tomuto účelu je do tohoto projektu implementována indexedDB. Jedná se o datové úložiště uvnitř prohlížeče. Tuto technologii však musí prohlížeč podporovat, v opačném případě nebude možno pracovat offline. Tato technologie je v projektu implementována tak, že si uchovává kompletní objekt přihlášeného uživatele, který díky svému návrhu udržuje veškeré potřebné informace o uživateli a jeho fakturách a zároveň tabulku veškerých provedených změn. V případě výpadku internetu, bude tento objekt neustále

aktualizován, a po opětovném navázání spojení budou změny odeslány do databáze, aby byla zajištěna konzistence dat.

Poslední, neméně důležitou součástí stránky je její vzhled. Vzhledem k tomu, že se jedná o webovou aplikaci, jsou pro definování vzhledu využity kaskádové styly. V celém projektu jsou pouze dva soubory s příponou CSS. Jeden slouží pro nastavení vzhledu úvodní stránky a druhý slouží pro nastavení vzhledu dashboardu. V momentě, kdy se uživatel přesune mezi těmito režimy, je vždy načten potřebný CSS soubor. Důvodem pro tuto volbu bylo logické oddělení těchto dvou vzhledově odlišných částí webové aplikace.

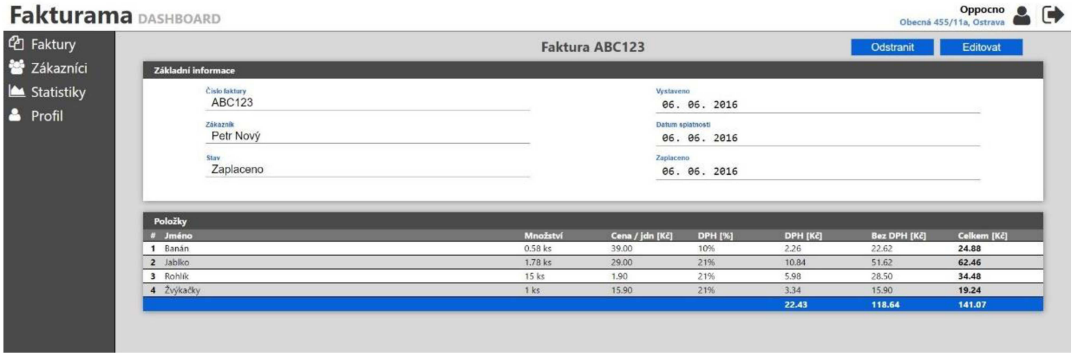
Samotný vzhled úvodní obrazovky je laděný do odstínů šedé barvy. Formuláře jsou dále doplněny barvou modrou. Popis jednotlivých polí formuláře je vždy nad tímto polem a je označen znakem "*", v případě, že vyplnění tohoto pole je povinné. Položky horní nabídky jsou za pomocí kaskádových stylů a animací vytvořeny tak, že po najetí myší se zobrazí jejich popis a ikona se zmenší.



Obrázek 1.4: *Ukázka formuláře na úvodní obrazovce*

Vzhled dashboardu je laděn do stejných barev jako úvodní stránka. Modrá barva je zde však zastoupena ve větším množství. Ikony horní i postranní nabídky mění barvu po najetí a zůstanou zvýrazněny v případě, že na ně uživatel klikne. Tato funkce zjednodušuje orientaci na stránce a upřesňuje, na které stránce se uživatel v danou chvíli nachází.

Všechny formuláře v dashboardu využívají stejného stylu. Jediné co se zde liší, jsou jejich popisy. Popisy jsou v případě, že není vstup vyplněn, nadepsány světle v pozadí vstupu. Po kliknutí do vstupního pole vyjede tento popis nad vstupní pole a tvoří menší nadpis tohoto pole. Informace o chybách se zobrazují pod každým polem zvlášť.



Obrázek 1.5: *Ukázka uživatelského rozhraní dashboardu*

Vzhled byl celkově navržen tak, aby byl přehledný a jednoduchý, zároveň neodebírá žádné funkce systému. V následující kapitole jsou prezentovány výsledky testu navrženého uživatelského rozhraní.

5 Uživatelské testování

Poslední částí této práce je testování uživatelského rozhraní. Vývojář, který systém a jeho uživatelské rozhraní navrhuje, nemůže odhalit každou chybu v uživatelském rozhraní. Z tohoto důvodu se využívá uživatelské testování, které pomáhá odhalit chyby, které během vývoje nebyly nalezeny.

V rámci tohoto testování byl předem připraven úkol, který byl sestaven z několika dílčích kroků. Tento úkol byl představen třem uživatelům. Uživatel A – žena, 22 let, zkušený uživatel PC, uživatel B – žena, 12 let, nezkušený uživatel PC, uživatel C – muž, 41 let, pokročilý uživatel PC.

Úkol, který byl zadán uživatelům, se skládal z těchto kroků: registrace uživatele, přihlášení uživatele, editace údajů o uživateli, přidání zákazníka uživatele a vytvoření jednoduché faktury pro tohoto zákazníka.

Prvním krokem úkolu byla registrace. Pro uživatele A nebyl větší problém najít ikonku pro přihlášení naopak pro uživatele B, bylo velmi náročné se na úvodní stránce zorientovat a najít místo, kde se může vůbec registrovat. Vzhledem k tomuto zjištění by bylo lepší změnit ikonku pro přihlašování a zároveň změnit název „Login“ na „Přihlásit se“. Při registraci vznikl problém pro uživatele C, jelikož vložil neplatný vstup, jak pro email, tak pro heslo. Vyhodila se chybová hláška „neplatný vstup“, ovšem uživatel C nevěděl, kde udělal chybu, obzvlášť v případě hesla, z toho důvodu by mělo být upraveno upřesnění vstupních podmínek po zadání špatného vstupu. Všechny ostatní kroky během registrace již proběhly bez problémů.

Druhým krokem bylo přihlášení uživatele, kdy se uživatel po registraci musel přihlásit na svůj nově registrovaný účet. Uživatel A ověřil, zdali nelze provést přihlášení bez aktivačního emailu, což by byla chyba. Po testování ovšem uživatel A zjistil, že musí být použit aktivační odkaz. Z tohoto plyne, že přihlášení funguje podle stanoveného plánu. Další komplikace u žádného z uživatelů nebyly zjištěny.

Třetím krokem byla editace údajů o uživateli. K editaci těchto údajů se lze dostat dvěma způsoby. Pro uživatele B i C bylo jednodušší najít editaci přes levé menu v sekci „Profil“. Uživatel A zvolil pro editaci ikonku osoby ve vrchní části stránky. Uživatel A odhalil, že kontrola vstupu pro registrační formulář, není plně funkční při registraci, ale až při editaci profilu, konkrétně v případě PSČ. V editačním formuláři je již vyřešen problém popsání chybných vstupů. Editační formulář vracel očekávané výsledky všem uživatelům

Přidání zákazníka bylo čtvrtým krokem. Tento krok byl zhodnocen všemi uživateli jako nejjednodušší a nejpřehlednější. Vytvoření zákazníka bylo podle očekávání nalezeno pod sekcí z levého bočního menu s názvem „zákazníci“, poté již stačilo kliknout na tlačítko „+ Nový zákazník“. Ošetřené vstupy byly také správně popsány, když došlo k nesprávnému vstupu.

Posledním krokem tohoto testování bylo vytvoření jednoduché faktury, pro zákazníka, jež byl vytvořen v předchozím kroku. Nalezení místa, kde se faktura vytváří, bylo jednoduché pro všechny uživatele jako v předchozím kroku. Uživatel C zvládl bez problému vyplnit všechny položky, jak v základních informacích o faktuře, tak v případě položek na faktuře. Uživatel B měl menší problémy s orientací u tabulky „Položky“, což přisuzuji nezkušenosti s touto problematikou uživatele. Uživatel A měl pár připomínek v rámci korektnosti fakturačního systému. Hlavní připomínka se týkala číselného označení faktury, které musí být unikátní a zároveň musí za sebou navazovat. Tento vstup není nijak ošetřen, proto mohl uživatel A vytvořit dvě faktury s totožným číslem, což je v reálném případě

nepřípustné. Uživatel A dále uvedl, že by bylo vhodnější místo v tabulce „Položky“ uvést místo „jméno“ označení „název“.

Z pozorování chování jednotlivých uživatelů v rámci plnění zadaného úkolu, jsem došel k závěru, že je fakturační systém přehledný a lehce ovladatelný i pro nezkušené uživatele. V rámci testování bylo odhaleno několik chyb, které se týkají, jak samotné funkcionality, tak správnosti při vytváření faktur. Díky tomuto testování mohou být opraveny chyby a systém může být dále vyvíjen, což je velké pozitivum uživatelského testování.

Závěr

Cílem této bakalářské práce bylo vytvořit vlastní fakturační systém vytvořený jako webová aplikace. Tohoto cíle se podařilo dosáhnout za použití mnoha různých technologií, jak pro tvorbu webových aplikací, tak pro tvorbu serverové části aplikace. Prvním krokem, který výrazně usnadnil návrh samotného fakturačního systému, bylo srovnání již existujících řešení. Pomocí tohoto srovnání pak bylo snadné vybrat funkce, které by měl tento fakturační systém obsahovat. Další funkcionalita fakturačního systému byla přímo zadána a jednalo se o možnost práce bez přístupu k internetu.

Pro vytvoření tohoto fakturačního systému bylo následně třeba navrhnout vlastní model databáze pro všechny potřebné objekty. Vytvořit pro něj také komunikační a prezentační vrstvu. Pro potencionálního uživatele je stěžejní právě prezentační vrstva, kterou lze celý systém jednoduše ovládat.

Toto uživatelské rozhraní bylo podstoupeno zkoušce několika náhodnými uživateli, kteří dostali za úkol odpovídat na předem zadané otázky, a podle výsledků bylo možno celkově zhodnotit výsledný fakturační systém.

Vzhledem k tomu, že se jedná pouze o bakalářský projekt a po porovnání výsledného fakturačního systému s jinými již existujícími bylo dosaženo závěru, že není vhodný pro použití pro komerční účely. Bylo by především nutné zvýšit bezpečnost této aplikace a přidat několik funkcí, které většina ostatních fakturačních systému také nabízí. Celkově se tedy jedná o velice zajímavou zkušenost, za pomocí které lze vytvářet zajímavější a propracovanější webové aplikace. Žádný další vývoj tohoto fakturačního systému již není plánován.

Použitá literatura

[1] Faktura. Wikipedia. [online]. 4. 4. 2016 [cit. 2016-04-19]. Dostupné z: https://cs.wikipedia.org/wiki/Faktura

[2] Náležitosti faktury a co musí faktura obsahovat?. fakturman.cz. [online]. © 2012 [cit. 2016-04-19]. Dostupné z:http://www.fakturman.cz/informace/nalezitosti-faktury-a-co-musi-faktura-obsahovat

[3] PILGRIM, Mark. HTML5: up and running. Sebastopol, CA: O'Reilly, 2010. ISBN 0596806027.

[4] MAHARRY, Daniel. TypeScript revealed. California: Apress, 2013. Expert's voice in .NET. ISBN 1430257253.

[5] TypeScript dokumentace. Typescriptlang.org [online]. 2016 [cit. 2016-06-28]. Dostupné z: http://www.typescriptlang.org/docs/tutorial.html

[6] J. D. GAUCHAT a EDITED BY JESSIE COLGAN. HTML5 for masterminds: [how to take advantage of HTML5 to create amazing websites and revolutionary applications]. 2nd ed. Vancouver: Mink Books, 2012. ISBN 1481138502.

[7] POWELL, Thomas A. Ajax: the complete reference. New York: McGraw-Hill, c2008. ISBN 007149216X.

[8] CROCKFORD, Douglas. JavaScript: the good parts. Sebastopol: O´Reilly, 2008. ISBN 978-0-596-51774-8.

[9] MACDONALD, Matthew. HTML5 the missing manual. Second edition. Sebastopol: O'Reilly Media, 2013. ISBN 9781449373443.

[10] Nový občanský zákoník pro každého. Bratislava: DonauMedia, 2014. ISBN 978-80-89364-55-8.

Seznam příloh

Příloha na CD.

Příloha A: *Entitně-relační diagram databáze*